Thesis for the Master's degree in Informatics

Yun Huang

Anatomic modelling of liver motion during the respiratory cycle

60 study points

Department of Informatics
Faculty of mathematics and natural sciences
UNIVERSITY OF OSLO 02/2008

Anatomic modeling of liver motion during the respiratory cycle

Yun Huang

Abstract

In this thesis, we present the methods for detect the liver motion during the respiratory cycle. We acquired 3D MR image sets of the liver of one volunteer at six time point in the respiratory cycle. By using a software package, 3D Slicer, we did semi-auto segmentation to extract the liver from the background image; and then generated surface model of the liver. The global motion of the liver is computed by rigid registration. The non-rigid deformation fields between the images were computed by using non-rigid registration algorithm. In addition, we visualized the liver models in 3D, and made an animation to show how the liver moves.

Thesis supervisor: Eigil Samset, Petter Risholm

Acknowledgments

In this section I will thank those who gave me support and encouragement.

I thank my supervisor Petter for his support at the beginning of my work. I thank my supervisor Eigil, who gave me so much useful advices and spent much time to go through my thesis and correct my English. And thank my family, none of this would be possible without their strongly encouragement.

Contents

List of Figures

.

Chapter 1

Introduction

This chapter presents the background and importance of this study, states the main motivation of the study described in the thesis, and gives a problem description. At the end of this chapter an overview of previous work in this area will be given.

1.1 Background

Image-guided surgery (IGS) is a broad term covering a wide range of surgical procedures where the surgeon uses indirect visualization to operate, i.e. by employing imaging instruments in real time, such as fiber optic cameras, internal video cameras, flexible or rigid endoscopes, ultrasonography, etc [1]. A real-time correlation of the operative field to a preoperative imaging data set needs to be established, allowing a visualization system to show the precise location of a selected surgical instrument in relation to the surrounding structures. Most image-guided surgical procedures are minimally invasive.

Computer-aided surgery is a kind of image-guided surgery where the surgical procedures are conducted with the aid of computers. It combines medical imaging (e.g. CT scans or magnetic resonance imaging (MRI)) with instrument tracking (e.g. optical tracking using infrared cameras) to guide surgeons and help them navigate to specific targets in the body with the help of three-dimensional images. Image-guided surgery is minimally invasive, with high precision and is becoming a tendency for the future operation. Image-guided procedures can often be divided in two phases:

- Before surgery, a pre-operative imaging data set is taken of the patient with either a CT or an MRI scanner.
- During the surgery, a tracking target is fitted to a surgical tool.

The cameras tracking these targets are connected to a computer. The three-dimensional images are reformatted in real-time and superimposed with information provided by the camera. The navigation system can be manipulated by the surgeon using a touch-screen or mouse. Figure 1.1 shows a navigation system for neurosurgery.

Figure 1.1: Image-guided surgery system for navigation to anatomical targets registered on pre-acquired images. [28]

1.2 Motivation

Although pre-operative images provide essential anatomical information for operative planning and help surgeons to estimate the position of internal organ target during operation, these images can not be used directly for precise localization during surgery. A registration process will be used to map the location of a set of points in the patient to the same points in the images. After registration, given that the organ is not moved, the internal target can be targeted precisely. However abdominal organs move during respiration. When the patient is under general anesthesia the respiration can be stopped for shorter periods of time. When local anesthesia is used, the patient can be instructed to hold the breath. However, this is not a reliable technique, and cannot be assumed to produce repeatable results. A number of algorithms have been developed and several groups are currently addressing the problem of movement of abdominal

organs. This has applications in imaging as well as image-guided surgery. Being able to compensate for this motion requires a thorough analysis of the motion in question. In this work we have investigated how the liver moves during respiration based on MRI imaging. The results data can be a reference to the development of real-time registration algorithms.

1.3 Problem description

The liver is an abdominal organ that moves and deforms during the respiratory cycle [2][3][4]. Because of its large size, the motion and deformation is more defined than other organs. The research subject of this project will be the liver motion.

One way to show the motion and deformation intuitively is to generate an animated 3D model of the liver which describes the temporal change in position and shape over time.

It would be optimal to acquire images of the liver every microsecond during the respiratory cycle, so we can catch all transformations and deformations. However, there is no machine that can scan a whole liver with adequate speed and when using breath-hold it is not possible to catch the respiratory condition. Since we can not get very dense sampling of the motion of the liver under respiration, we have to make some compromises and rather aim for a few respiratory data points which are easy to control, and find the motion between these points by using interpolation and approximation to get the whole motion. It is important to make sure that the motion does not change direction.

To define the objectives more clearly, we want to first get the real location of the liver in a given respiratory phase during the respiratory cycle. By comparing these data we want to catch both globe translation and local deformation of the liver during the respiratory cycle. From the globe translation we want to get some informations about the movement of liver at every direction in a 3D

environment. And we are expected to find out the deformed region in the liver, and to understand how it deforms under the respiration.

1.4 State of art

A number of studies have already been performed to understand the motion of the liver during respiration. These studies suggest that clinically significant liver motion can be approximated by superior-inferior movement alone with relatively minimal motion along the other axes. They also show that deformation is present.

1.4.1 Korin [2]

.

This paper (Korin et al., 1992) describes an experiment using MR line scans to measure the relative displacement of the liver during breathing. Line scan is a one dimensional imaging technique. The measurement technique consisted of spin-echoes derived from spatially localized lines of tissue defined by the intersection of orthogonal slice-selective 90 degree and 180 degree rf pulses. The pulse sequence and imaging geometry for the specific case of a coronal 180 degree plane and a sagittal 90 degree plane. This yields a line scan along the longitudinal direction. One-dimensional Fourier transformation of each line scan echo provides a measure of object position along the scanned line. Juxtaposition of such transformed echoes acquired successively for the same line provides a display of how each object moves as a function of time. The experiment included imaging 15 volunteers during both normal and deep breathing. The average ratio of superior-inferior (S-I) motion of the diaphragm versus right-left (L-R) motion of the abdomen wall was found to be 6.0 ± 1.9 for normal breathing and 8.8 ± 3.8 for deep breathing. The average ratio of S-I motion of the diaphragm versus anterior-posterior (A-P) motion of the diaphragm wall was 5.1 ± 1.9 for normal breathing and 5.0 ± 2.7 for deep breathing. The average

amplitude of S-I translation was 1.3cm for normal breathing and 3.9cm for deep breathing. They investigated deformation of the liver by acquiring a coronal image through the liver at normal exhale and then normal inhale. To measure the dilation, a coronal breath hold image was acquired at normal inspiration and then at normal expiration. Two vessels near the top and bottom of the liver were used as markers to provide a measure of dilation between the top and bottom of the liver. They observed that a dilation in S-I direction is less than 3% at exhale relative to inhale. The authors conclude that the motion of liver is mainly in S-I direction and also little deformation.

1.4.2 Rohlfing [3]

This paper (Rohlfing et al.), 2001, describes a technique for modelling liver motion during the respiratory cycle the author used based non-rigid registration of gated MR images.

The abdominal MR images from four volunteers were acquired on two MR scanners, and acquired eight images between inhale and exhale using respiratory gating on each of all subjects. The state of end-expiration image is generally the most reproducible phase in the respiratory cycle [4], this image was used as the reference frame for all registrations.

A non-rigid registration algorithm used, employing a free-form deformation defined on a discrete uniform control point grid with cubic B-spline approximation between adjacent control points. By applying the registration algorithm, seven transformations were computed that mapped images in seven phases in the respiratory cycle with respect to end-expiration image. Rigid and non-rigid registrations were then performed by inspecting a variety of image fusions, including iso-intensity contour overlays, checkerboard fusion, and image subtraction. The result shows that the dominant component of liver motion was cranial-caudal translation. This translation between inspiration and expiration ranged from 12 to 26mm. Translation in the anterior-posterior

direction were between1 and 12mm and in the lateral direction were between 1 and 3 mm. The non-rigid transformation was found inside the liver and was high as 19mm for one volunteer. For certain regions of the liver, the location during the respiratory cycle was almost 2cm away from where rigid registration would have predicted it to be. The average difference between inspiration-expiration over the entire liver in all four volunteers was approximately 6 mm.

1.4.3 Wong [5]

This paper (Wong et al., 2004), describes an experiment through which they wanted to find the relationship between the motion of the skin and the motion of the liver. The aim was to be able to use skin motion as a predictor for complex internal target motion.

Volunteers were scanned under both breath-hold and free-breath conditions.

Four potential MR pulse sequences were tested: (a) T1 weighted in- and out-of-phase images, (b) T2 weighted half-Fourier single shot turbo spin-echo (HASTE), (c) T2/T1 weighted fast imaging with steady-state precession (trueFISP), and (d) 3D fast acquisition multi-echo (FAME).

For some volunteers, the researchers tested prospective respiratory gating based on the respiratory bellows to ensure that the images were derived from known points in the respiratory cycle. The outline of the liver was segmented. The liver position and orientation were compared between end-inspiration and end-expiration. The non-gated images acquired under free breathing were first pre-processed to sort the files according to their acquisition time, so that the slices could be binned into their approximate location in the respiratory cycle. This binning generated 4- 6 image sets reflecting the anatomy at different time points. Analysis of these images then proceeded as with the breath-hold images. This method used breath-hold images as a “baseline” for comparison against the images taken during normal breathing. Their conclusion is that the movement of the skin varied depending on the individual and the anatomical location, and

ranged between 0 and 1.6 cm. Liver motion was most significant in the cranial-caudal direction; displacements as large as 4.4 cm were observed in breath-hold images, whereas under free breathing the largest displacement was 3.2 cm.

1.4.4 Shimizu [6]

This paper (Shimizu et al., 2000) describes the measurement of tumour movement in the liver due to respiration using MR imaging and reference markers. A marker, which has five parallel bars in every 2 cm, was placed on the surface of the patient as a ruler for measuring purpose. Each bar can be seen as a dot on a MR image, when they are scanned vertically to their axis. A phased array body coil was set up on to the surface of the patient, to give as little restriction of the respiratory movement of the abdominal wall as possible. Repetition time was 7.7 ms, echo time was 4.2 ms, flip angle was 20°, section thickness was 8 mm, and a 256×128 matrix was used. The acquisition time was 1.0 s followed by an interval of 0.5 s. The 20 tumor contours extracted during 30 s were superimposed on sagittal and coronal MR images. The result was as follow: The maximum value of tumor edge location was 3.9 cm in the cranio–caudal direction, 2.3 cm in the ventro–dorsal direction, and 3.1 cm in the lateral direction. The mean length of tumor displacement observed was 2.1 cm in the cranio–caudal direction, 0.8 cm in the ventro–dorsal and 0.9 cm in the left–right direction, respectively. The locus of the center of the tumor contour in the sagittal cross section was inclined at 23° and in the coronal cross section was inclined at 18° to the cranio–caudal axis of body.

1.4.5 Balter [7]

In this paper (Balter et al. 1996) attempted to assess the errors in radiotherapy planning by using CT images, which were acquired under shallow and free breathing. For each of six patients with abdominal tumors, they acquired three images: One at normal inhalation, one at normal exhalation, and one while the

patient was breathing. After looking at the change in position of the centre of mass of the liver the got following result: Inhalation and exhalation data differ in terms of radiation path length (nearly one quarter of the cases had path-length differences >1 cm), although the free breathing and average path lengths do not exhibit large differences (0–9 mm). Liver and kidney movements averaged 2 cm, whereas differences between the free breathing and average positions averaged 0.6 cm.

From these researches we can conclude that:
The motion of liver includes both rigid formation and non-rigid formation. The dominant component of liver motion was cranial-caudal translation and is from 12 to 44mm. A few key areas seem most prone to deformation, the most intuitive of which is the superior surface of the liver, which is in direct contact with the diaphragm and so will experience a large downward force during inhalation. Another region in which large deformation may be observed is the inferior surface of the liver, where it is in contact with the back of the body cavity, and is compressed against this surface as the diaphragm' descends. Some deformation is often seen around the large blood vessels in the liver. Regions around the inferior vena cava, particularly where it passes through the diaphragm, or at the level of the main portal vein may experience compression and thereby expel blood, closing the vessels to a certain extent. The same seems to apply for other major vessels within the liver.

Chapter 2

Method

This chapter describes the experiment method that was used in this study. The process pipeline was as follows:

Image acquisition – Images were acquired by scanning a volunteer with Magnetic Resonance scanner. The volunteer was scanned in six respiratory phases. This will be presented in chapter 2.1.

Image preparation – A free software toolkit, 3D Slicer, was used to load the image-sets. This will be presented in chapter 2.2.

Image segmentation – Distinguishing the liver from the background. Threshold and drawing contour lines will be presented in chapter 2.3.

Model generation – Generating the geometric liver models derived from the sampling data. This will be presented in chapter 2.4.

Model Visualization – Visualizing the liver models and making animation to show the movement of the liver. This will be presented in chapter 2.5

Image registration – Transforming the different sets of data into one coordinate system. This will be presented in chapter 2.6 and 2.7

2.1 Image acquisition

In this chapter we are going to introduce the process of respiration, and then discuss at which time points in the respiratory cycle we performed imaging. At the end of this chapter we will represent how the images were acquired.

2.1.1 Expected images

In order to accomplish our goals, we want to generate a 3D model of the liver which describes the temporal change in organ position and shape during respiration. In this experiment we choose Magnetic Resonance Imaging (MRI) to represent the whole respiratory cycle. MRI is used to diagnose diseases and

abnormalities in all parts of the body and provides detailed images of the body in any plane and can reveal detailed internal structures.

Before acquiring the MRI images, we have to determine which phase of the liver under respiratory cycle should be used. The sampling points should have the following characteristics:

1. Easy to perform. The person being scanned should understand clearly which state of respiration he should perform. And it should be easy for him to control his breath in order to reach the respiratory state we want to scan.
2. Cursorily represent the whole breath cycle. These points should not be many, but each point should represent a characteristic respiratory phase. Few points should be easy for us to reproduce the respiratory cycle. And by finding out the liver motion at these characteristic points, we can simply use some interpolation and approximation method to predict the motions between these points.

2.1.2 Respiratory cycle

To choose which points in the respiratory cycle should be used, we have to understand the process of the respiratory cycle.

[10] [11] Respiration is the movement of air into and out of the human body. People breathe with the help of the diaphragm (diaphragm is a dome-shaped muscle under the rib cage) and other muscles in the chest and abdomen. These muscles will literally change the space and pressure inside the chest cavity to accommodate the breathing. During inspiration the lungs increase in volume with air, the pressure inside the lungs becomes negative compared to the atmospheric pressure. Due to the negative pressure, air flows into the lungs. At the same time the diaphragm pulls down to make room for the lungs to expand. When the lungs remain increased at a constant volume, the negative pressure disappears and the flow ceases. During this process the lungs will expanded by air like two balloons being blown up. During expiration, the lungs decrease in volume, the pressure inside the lungs is positive compared to the atmospheric

pressure. Due to the positive pressure, the gas inside the lungs flows out. The diaphragm relaxes, it moves up and the cavity inside the body gets smaller. The muscles will then squeeze the rib cage and the lungs begin to collapse as the air is pushed up and out the body. When the lungs remain decreased at a constant volume, the positive pressure disappears and the flow ceases. The process of inspiration and expiration is shown in Figure 2.1 and Figure 2.2.

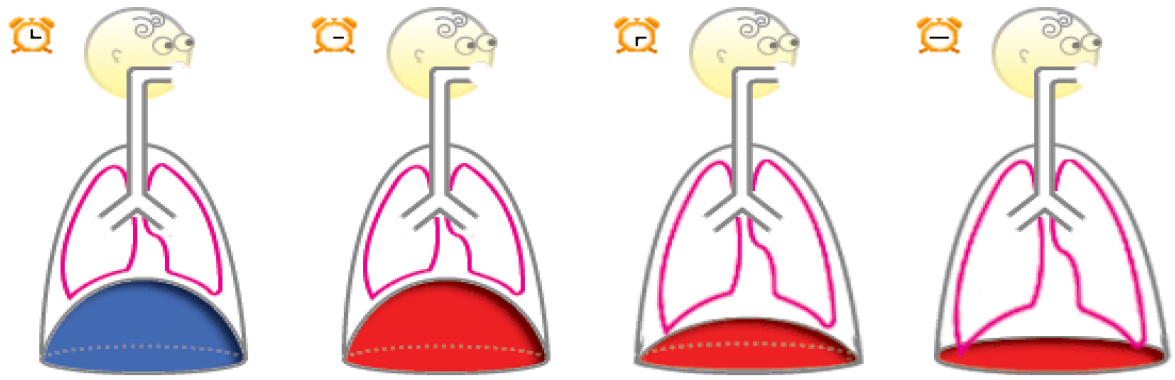


Figure 2.1: The process of inspiration [9]

The red outline shows the lungs, the outer grey line shows the chest wall, the cavity inside the chest wall is the thoracic cavity; the diaphragm is at the bottom of the thoracic cavity.

1. At the beginning of inspiration, the diaphragm is contracting; the contents of the abdomen are beginning to move downward, the lung volume is at its minimum.

2. 3. During inspiration, the diaphragm is contracting; the lung volume is increasing from its minimum to its maximum.

4. At the end of inspiration, the diaphragm is contracting; the lung volume is at its maximum.

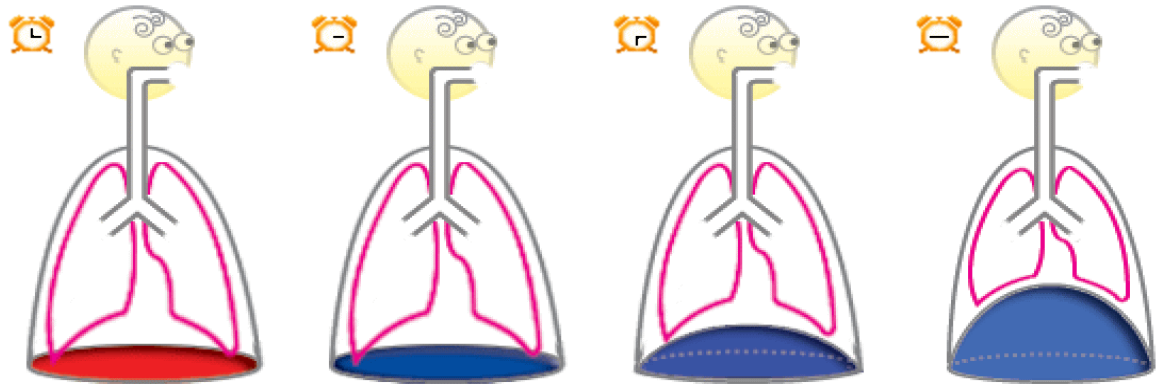


Figure 2.2: The process of expiration [9]

The red outline shows the lungs, the outer grey line shows the chest wall, the cavity inside the chest wall is the thoracic cavity; the diaphragm is at the bottom of the thoracic cavity.
1. At the beginning of expiration, the diaphragm is relaxing; the contents of the abdomen are beginning to move upward, the lung volume is at its maximum.
2. 3. During expiration, the diaphragm is relaxing; the lung volume is decreasing from its maximum to its minimum.
4. At the end of expiration, the diaphragm is relaxing; the lung volume is at its minimum.

According to the process of inspiration, there are four points which are interesting for us: At the beginning of inspiration, where the diaphragm is contracting, and the contents of abdomen are beginning to move downward; at the end of inspiration, where the diaphragm is contracting to its limit, and the volume of the abdomen is at its minimum; at the beginning of expiration, where the diaphragm is relaxing and the contents of the abdomen are beginning to move upward; at the end of expiration, where the diaphragm is relaxing to its maximum, and the volume of the abdomen is also at its maximum. As we know the inspiration and expiration is a continuous cycle, the point at the end of inspiration and the point at beginning of expiration is the same; the point at the end of the expiration is the same as the point at the beginning of inspiration. Obviously, it is not enough to have only two sampling points. In addition, it was indicated in the early research [3], that the local deformations of liver could be very large, and we can not easily capture them with only two points. So in this experiment we will choose two extra points during both inspiration and expiration. These four points are: at the middle of inspiration; near the end of inspiration; at the middle of expiration; and near the end of expiration. As we are going to mention these points during the whole experiment, we call these points I_0, I_1, I_2, I_3, I_4 and I_5 (Figure 2.3).

2.1.3 Image acquisition

The MRI images were acquired by placing a volunteer in a MRI scanner. The volunteer is a health young man at age about 20. The reason of choosing a young healthy person is that our research object is an abdomen organ and any disease may affect us to get reasonable results. Another reason is that it takes time to scan people under breath hold, so the person who is been scanned should be able to suspend his breath over 20 seconds, and a young healthy person can do this easily. The volunteer was positioned in the scanner in such a way that the entire liver would be imaged at all positions and was scanned in a supine position while suspending his breathing at the six points in the respiratory cycle: I_0, I_1, I_2, I_3, I_4 and I_5.

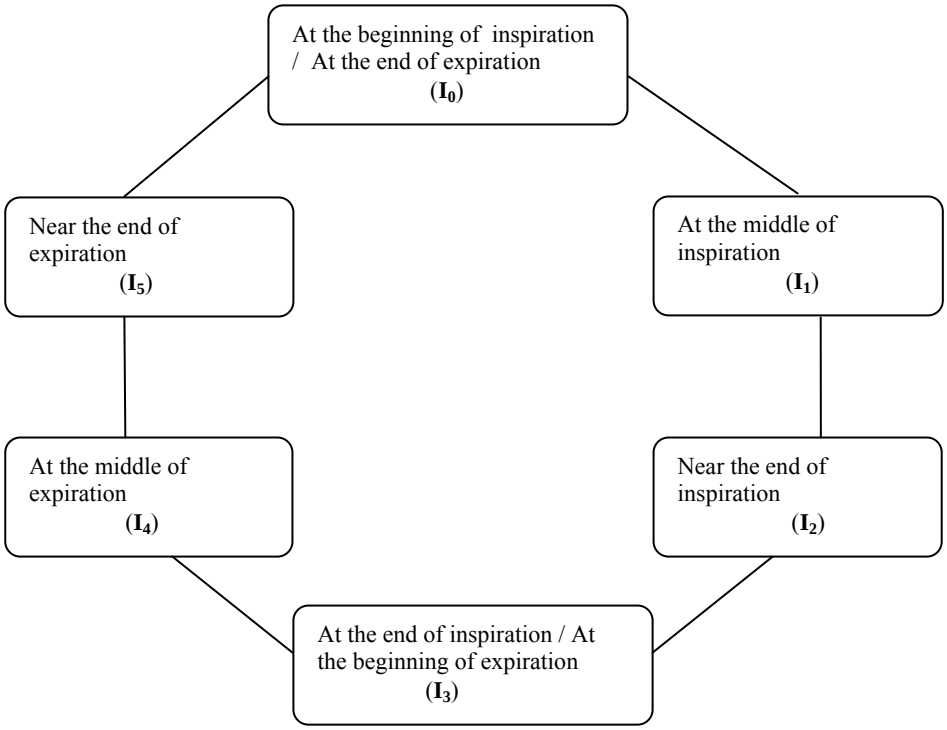


Figure 2.3: The sampling points in respiratory cycle.

The main component of a MR scanner is a large cylindrical magnet. The scanner uses magnets and radio waves to create cross-sectional images of the human body. [8][9] The process that lies behind the acquisition of an MR image can be explained as follows:

Water molecules are dipoles and behave like small magnets. When these small magnets are placed in a magnetic field, they align with the field. If a radio frequency (RF) pulse is transmitted at a given frequency (proportional to the magnetic field strength), the dipoles will start to spin around the magnetic field. When the RF pulse is turned off, the dipoles will return to their equilibrium, where they are aligned with the magnetic field. In this process some of the energy of the dipoles will be lost, and some of it can be picked up by a receiver coil and used to generate MR image.

An MR system is consisting of three major parts: Magnet and coils, electronics and a computer system. When the body lies in a magnet, it becomes temporarily magnetized. This state is achieved when the hydrogen nuclei in the body align with the magnetic field. When magnetized, the body responds to exposure to radio waves at a particular frequency by sending back a radio wave signal called a "spin echo". This phenomenon only occurs at one frequency corresponding to the specific strength of the magnetic field. The spin echo signal is composed of multiple frequencies, reflecting different positions along the magnetic field gradient. When the signal is broken into its component frequencies, the magnitude of the signal at each frequency is proportional to the hydrogen density at that location, thus allowing an image to be constructed.

The MR scanner and its directions are shown in Figure 2.4. Throughout this thesis anterior-posterior refers to the Y direction will be abbreviated to A-P, superior-inferior (S-I) refers to the Z direction and right-left (R-L) refers to the X direction.

After scanning we got six set MR images which represent the six points in the respiratory cycle.

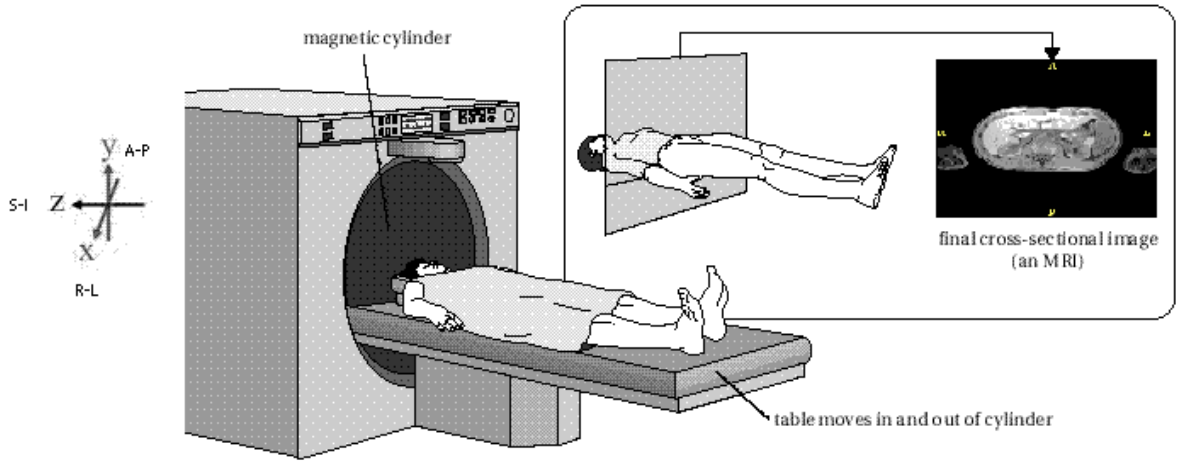


Figure 2.4: MRI scanner and directions.

2. 2 Image preparation

Once the MR images were acquired, we want to understand the information they contained and to review these images by some software, so that we can continue our study by working on them. In this chapter we will first introduce the format of MR images (chapter 2.2.1, chapter 2.2.3). Then introduce the software we are going to use in our study (chapter 2.2.2). At last we are going to talk about "voxelization" which enable us to orient our 2D images in 3D space (chapter 2.2.4).

2.2.1 DICOM

The format used to store and transfer MRI images is called DICOM (Digital Imaging and Communications in Medicine). With the introduction of computed tomography (CT) followed by other digital diagnostic imaging modalities such as MRI in the 1970's, and the increasing use of computers in clinical applications, the American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA) recognized the need for a standard method for transferring images as well as associated information between devices manufactured from various vendors. ACR and NEMA formed a joint committee to develop the DICOM standard. This standard was developed

in liaison with other Standardization Organizations such as CEN TC251, JIRA including IEEE, HL7 and ANSI USA as reviewers.

DICOM is a comprehensive set of standards for handling, storing and transmitting information in medical imaging. The DICOM standard was developed based on the previous NEMA specification. The standard specifies a file format definition as well as a network communication protocol. DICOM was developed to enable integration of scanners, servers, workstations and network hardware from multiple vendors into an image archiving and communication system.

DICOM files consist of a header and a body of image data. The header contains standardized as well as free-form fields (see chapter 2.2.3).

2.2.2 3D Slicer

The DICOM file is binary encoded; it can not be read directly. So we have to use some software to read this kind of data. “3D Slicer” is such kind of software which can do the job for us. It integrates several facets of medical image computing into a single environment, provides capabilities for automatic registration, semi-automatic segmentation, surface model generation, 3D visualization, and quantitative analysis of various medical scans. 3D Slicer will be used throughout the whole study.

“3D Slicer” was originally developed at the Surgical Planning Lab at Brigham and Women’s Hospital and the MIT AL Laboratory. It was designed to help surgeons in image guided surgery, to assist in pre-surgical preparation, to be used as a diagnostic tool, and to help in the field of brain research and visualization. “3D Slicer” is built on the ITK and VTK programming toolkits developed as part of the Visible Human Project whose goal is to create full and detailed 3D models of male and female human bodies.

2.2.3 Header information

By loading the DICOM data from “3D Slicer”, the header of the DICOM file first be extracted by “3D Slicer” (Figure 2.5).

Image size: Number of pixels (see chapter 2.2.4) of the image in the x and y direction.

Pixel size: Size of each pixel in the x and y direction.

Slice thickness: The z dimension of the voxel (see chapter 2.2.4).

Scan order: IS = inferior to superior.

Scalar type: The data format of the pixel. Shot means 16 bit integer.

Slicer tilt: The tilt of gantry during an MRI.

Num scalars: The number of scalar components for each voxel. 1 means grey-level data.

Little endian: In little-edian architectures, the right most bytes are most significant.

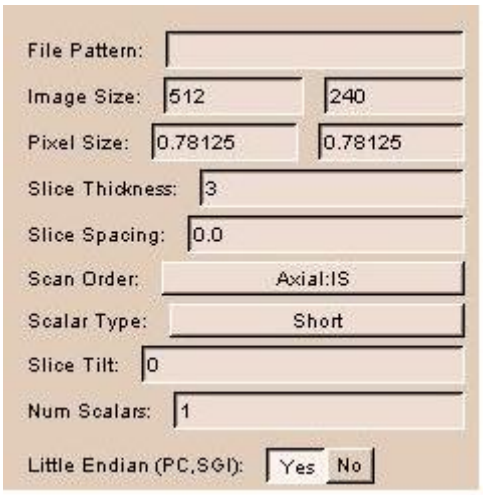


Figure 2.5: The header information of DICOM file.

2.2.4 Reformatting

Because the slices (MRI images) are made up of pixels, once the data have been converted into two-dimensional images, we have all the necessary information to create three-dimensional views of the object. [12] Pixel is the basic unit for the two-dimensional MR images. A pixel (picture element), is a single point in a

graphic image. The pictures are shown by dividing the display screen into thousands, or millions, of pixels, arranged in rows and columns. The pixels are so close that they connect each other. The number of bits used by the system to represent each pixel determines how many colours, or shades of grey, can be displayed. E.g. in 8-bit colour mode, the display monitor uses 8 bits for each pixel, making it possible to display $2^8 = 256$ different shades of grey.

After loading data into "3D Slicer", we got an interface as shown in Figure 2.6. The MRI data was scanned in the inferior-superior direction and the data is stored as a stack of 2D images. For 3D analysis these stacks of 2D images are combined to form a 3D volume.

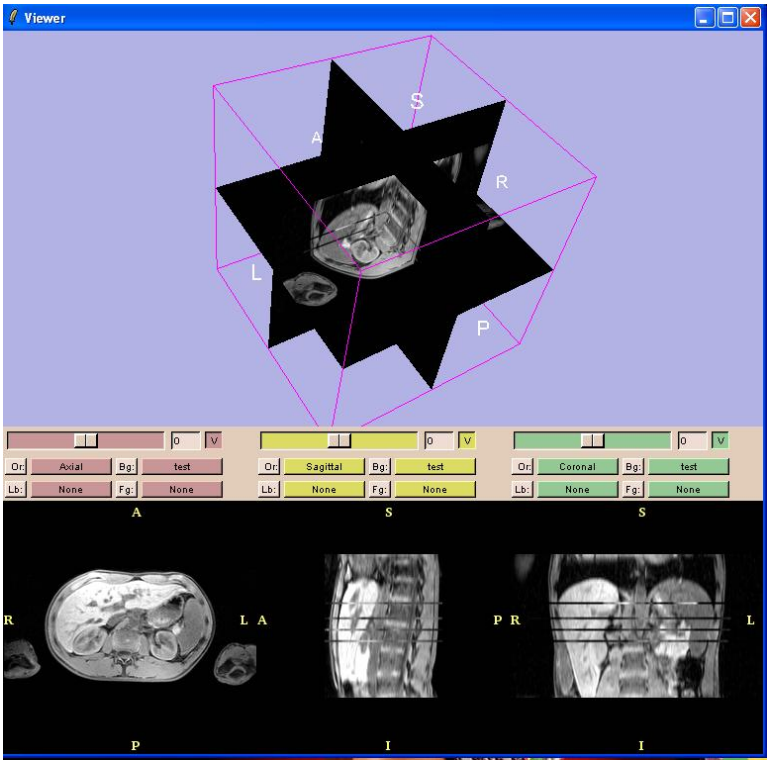


Figure 2.6: Interface in "3D Slicer".

Here we will first introduce the definition of voxel. Voxel is the basic unit in a 3D image grid. A voxel (volume element or volume cell) is the smallest distinguishable part of a three- dimensional image. It has three coordinates (x, y and z), which means that we have added depth to a pixel. The dimensions (volume) of a voxel are thus: 1 pixel width x 1 pixel high x 1 pixel depth. When

the two-dimensional MR images are put into a stack, the pixels take a new coordinate, the depth or *z*. Consequently the stack of images is converted to a three-dimensional image. [13] The process of adding depth to an image using a set of cross-sectional images, known as a volumetric dataset, is called voxelization. The slices are made up of pixels. The space between any two pixels in one slice is referred to as inter-pixel distance, which represents a real-world distance. And, the distance between any two slices is referred to as inter-slice distance, which represents a real-world depth. The dataset is processed when slices are loaded in computer memory based on inter-pixel and inter-slice distances to accurately reflect the real-world sampled volume. The additional slices are created and inserted between the dataset's actual slices so that the entire volume is represented as one solid block of data.

2.3 Data segmentation

Once the imaging data had been collected, the next and very important step was to segment the data. As shown in Figure 2.7, the MRI liver cross-sectional images show the liver along with other structures like the gallbladder and high intensity structure like backbones embedded on a dark background. Since our experiment's subject is just liver, we have to find some method to pick the liver out from the background. This process is called segmentation or classification. Segmentation is necessary for two reasons: first, to visualize the liver by means of surface model. Second, data registration (see chapter 2.6, chapter 2.7) algorithm will be implemented more accurately without disturbing of other tissues.

The process of liver segmentation involves separating and labeling the liver from anatomical structure images, so a 3D surface model can be generated on the segmented liver image and the segmented liver image can enable more accurate data registration.

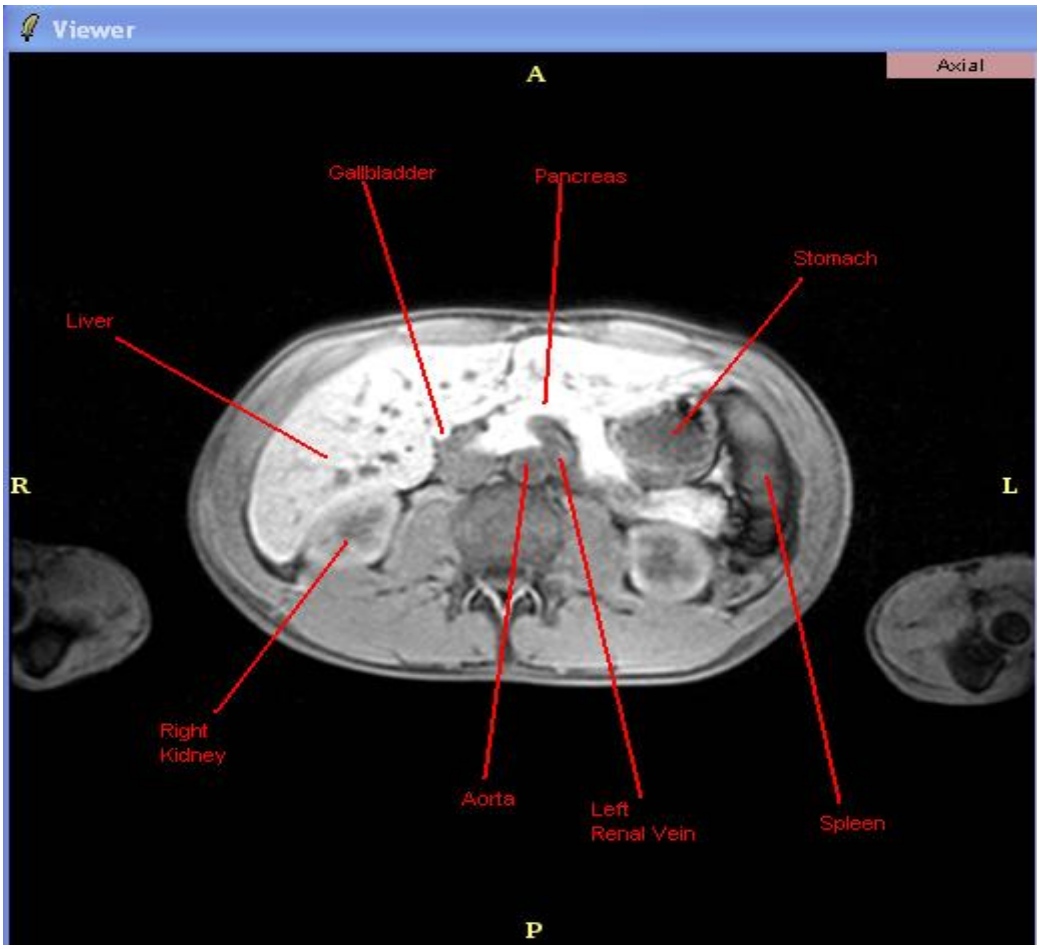


Figure 2.7: The liver and other structures in MR image.

"3D Slicer" includes different segmentation tools, such as threshold, connectivity, morphological operations, island removal, and free-hand drawing. Volumetric data can be segmented semi-automatically using these tools slice-by-slice on either axial, coronal, or sagittal slices.

In this study we used threshold and free-hand drawing on axial slices to segment data. Threshold was used to remove most of the noises and define a coarse contour of liver and then according to the defined contour (chapter 2.3.1), free-hand drawing was performed on a slice by slice basis to further refine the liver segmentation (chapter 2.3.2).

2.3.1 Threshold

Threshold is a commonly used tool in image segmentation when one is interested in identifying different homogeneous regions. It is very useful for

keeping the significant parts of an image and removes the unimportant part or noise. In our case, the interesting part of the images is the liver.

Threshold is an image processing technique for converting a greyscale or colour image to a binary image based upon a threshold value. Threshold selecting is essentially a pixels classification problem [14]. Its basic objective is to classify the pixels of a given image into two classes: one is those pertaining to an object and another is those pertaining to the background. While one includes pixels with gray values that are below or equal to a certain threshold, the other includes those with gray values above the threshold. In other words, if a pixel in the image has an intensity value less than the threshold value, the corresponding pixel in the resultant image is set to black. Otherwise, if the pixel intensity value is greater than or equal to the threshold intensity, the resulting pixel is set to white.

Many automatic threshold-determination techniques use the histogram of the image to select a good threshold. The histogram of an image is the frequency distribution of grey levels in that image. If, in an image, the objects have distinctly different grey values from the background, the histogram will exhibit different peaks with a valley between them (Figure 2.8).

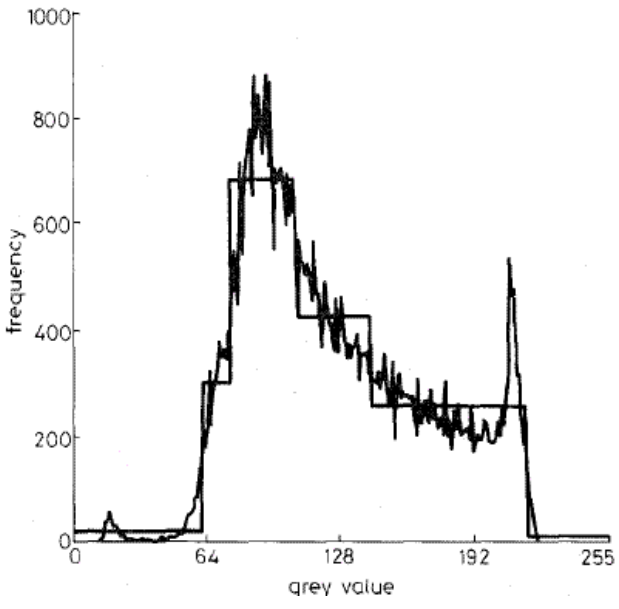


Figure 2.8: Threshold histogram [29].

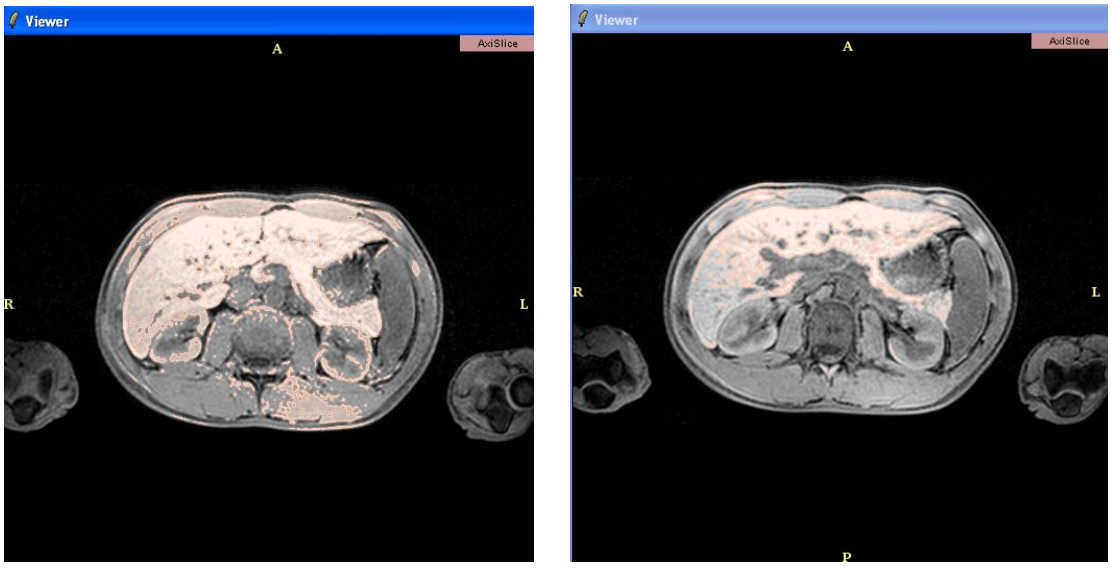
The threshold module in "3D Slicer" has several functions to automate using histogram. The selection of threshold values is done by adjusting the two sliders "Lo" and "Hi", a threshold value was applied to each slice image. Figure 2.9 shows results with different threshold value.

In Figure 2.9, the background is the original gray scale, and the foreground is the labelmap. The labelmap is a collection of labels that is produced through segmentation. The labelmap describes the class to which each voxel is assigned as an integer label according to the segmented structure it belongs to.

Example (a): Upper threshold is set at 294, and the lower threshold is set at 164. It does not remove all of the noise.

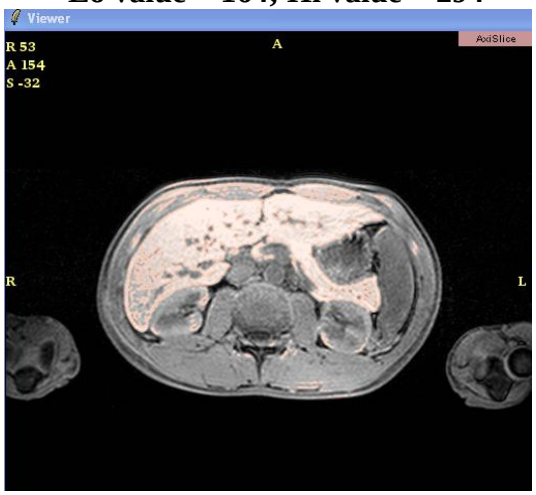
Example (b): Upper threshold is set at 294, and the lower threshold set is at 212. It does remvoe all of the noise, but also cuts too much around edge of liver.

Example (c): Upper threshold is set at 294, and the lower threshold is set at 188, which is a mean value between 164 and 212. It does remove a majority of the noise, and a clear contour is around liver. But same noise still exits. Those parts which have relatively low intensity inside liver present holes on the labelmap. To complete the segment the liver from the MR image, a manual segmentation has to be carried out.



a
Lo value = 164; Hi value = 294

b
Lo value = 212; Hi value = 294



c
Lo value = 188; Hi value = 294

Figure 2.9: Segmenting data by applying "Threshold" module.

2.3.2 B-spline segmentation

B-spline segmentation is a simple manual segmentation technique which is carried out by marking points around the liver contour getting by thresholding in each MR slice. B-spline curve is used to delineate the boundary of the liver on each slice in the data volume that contains the liver.

A B-spline (see chapter 2.7) of order k is a C^{k-2} continuous parametric curve that depends only on a number of control points $c_1, \ldots, c_n$.

A B-spline curve is an appropriate curve representation for deformable contour segmentation due to the following properties:

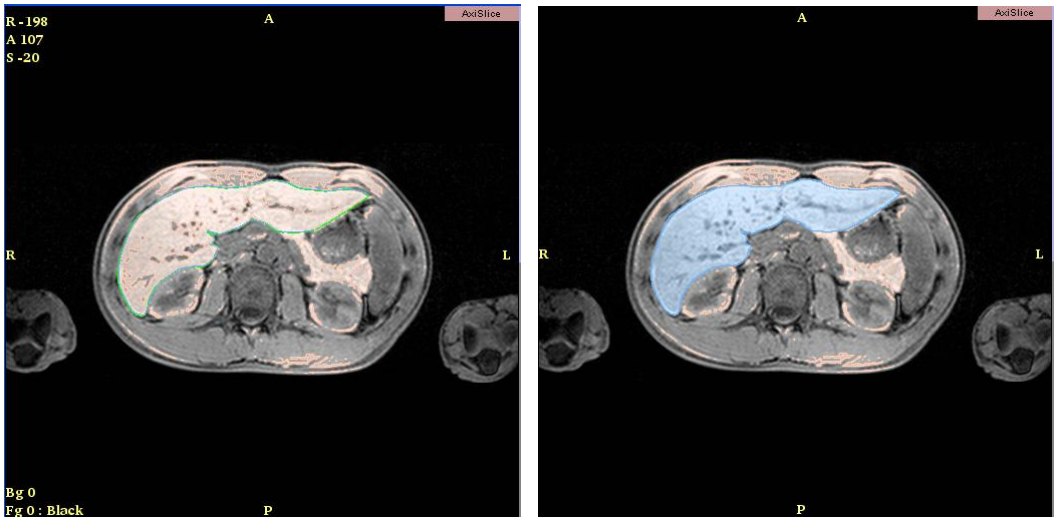
1) The smoothness of the curve corresponding to the internal energy of the contour can be adjusted by the number of control points.

2) The blending functions $B_i^k(u)$ have only local support, varying the position of one control point only affects a small part of the curve.

The shape of the B-spline curve is defined interactively, by moving its control points with the mouse in the image slice. It is also possible to add new control points or remove existing ones.

In “3D Slicer”, the segmentation tool free-hand drawing uses the B-spline algorithm. Free-hand drawing is involves marking points along the contour produced by threshold segmentation. Program automatically connected the points using B-spline curves (Figure 2.10 a). Since there exits only small difference between two adjacent slices and we know from the properties of B-spline curve that varying the position of one control point only affects a small part of the curve, instead of marking points along the contour of liver in every slicer, we can reduce our segmentation job by only add or move few points in the previous segmented slice. After setting B-spline curve along the contour of liver on each slice, a collection of label, labelmap is generated automatically by the program (Figure 2.10 b).

The process of segmentation in our study is to segment the six sets of MRI images one by one. We first employed threshold on the MR images to remove most of the noises and define a coarse contour. Secondly, we applied the tool “free-hand drawing” to draw B-splines around the liver contour slice-by-slice on axial direction to get a segmented liver.



a b

Figure 2.10: B-spline segmentation.
a. B-spline curve generated by drawing points along contour produced threshold segmentation.
b. Labelmap generated through B-spline segmentation.

The output of this segmentation process is six sets of label maps, where each voxel is assigned a value according to its membership in liver's anatomical structure. We call them l_0, l_1, l_2, l_3, l_4, and l_5. A label map is an intermediate step in creating a surface model, which is a set of joined triangles arranged in 3D space to form a closed surface.

2.4 Model generation

By performing segmentation, a surface model can be made from a label map by creating a mesh of triangles that bound all boundary voxels of a common label. The resulting triangle mesh (see chapter 2.4.1) is the 3D graphics equivalent of wrapping triangles around a liver.

The advantage of a surface model representation of the liver is that it gives a three-dimensional representation which can be rendered from any angle, which is an improvement over two-dimensional cross sections through the original grayscale data.

2.4.1 Marching cubes

When the bounding surfaces of the label maps are extracted, they can be represented as a collection of triangles using the Marching Cubes algorithm. The marching cubes (Lorensen and Cline, 1987 [15]), is the algorithm for rendering iso-surface in volumetric data. The basic principle behind the marching cubes algorithm is to subdivide space into a series of small cubes. The algorithm then instructs us to 'march' through each of the cubes testing the corner points and replacing the cube with an appropriate set of polygons. The sum total of all polygons generated will be a surface that approximates the one the data set describes.

The process of how marching cubes creates a surface from a three-dimensional set of data is as follows:

1) Create a cube from four neighbors on one slice and four neighbors on the next slice between two adjacent slices.

2) Calculate an index for the cube by comparing the eight density values at the cube vertices with the surface constant.

3) Since there are eight vertices at each cube and two states inside and outside, there are only 2^8=256 ways a surface can intersect the cube. By reflections and symmetrical rotations, the 256 cases can be reduced to 15 unique cases as shown in Figure 2.11[16]. We can then create an index for each case base on the state of the vertex. Using the index, look up the list of edges from a pre-calculated table.

4) Using the densities at each edge vertex, find the surface edge intersection via linear interpolation.

5) Calculate a unit normal at each cube vertex using central differences. Interpolate the normal to each triangle vertex.

6) Output the triangle vertices and vertex normal.

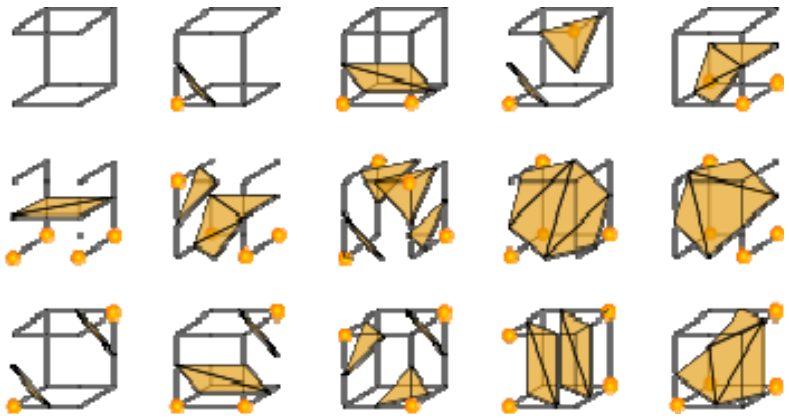


Figure 2.11: Cube combinations. [16]

The marching cubes algorithm produces a triangle mesh by computing iso-surfaces from discrete data. [17] By connecting the patches from all cubes on the iso-surface boundary, we get a surface representation in .vtk format.

In our experiment we got six surface models from the six set of label maps, and call them m_0, m_1, m_2, m_3, m_4, and m_5.

2.5 Model Visualization

The surface models describe the process of forming the shape of the objects. To extract the shape of the objects, we have to study the surface files we got. The surface files are vtk formatted, which can be read by VTK (chapter 2.5.1). After reading file (chapter 2.5.2), we convert the surface models into images and rendering them onto the screen (chapter 2.5.3).

2.5.1 File format

The files we got look like as Figure 2.12. All data are written in binary form.

```
# vtk DataFile Version 3.0
vtk output
BINARY
DATASET POLYDATA
POINTS 60968 float
.
.
.
TRIANGLE_STRIPS 25401 197365
.
.
.
POINT_DATA 60968
NORMALS Normals float
.
.
.
```

Figure 2.12: VTK file format.

To read such data, we have to understand how the file is constructed. First we are going to introduce VTK.

2.5.1.1 VTK

The Visualization ToolKit (VTK) [23] is a freely available software system for 3D computer graphics, image processing, and visualization. VTK is a C++ class

library, and also includes several interpreted interface layers including Tcl/Tk, Java, and Python. VTK has been implemented on nearly every Unix-based platform, PC's (Windows 95/98/NT/2000/XP) and Mac OSX Jaguar and later. VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods; and advanced modeling techniques such as implicit modelling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation. In addition, dozens of imaging algorithms have been directly integrated to allow the user to mix 2D imaging / 3D graphics algorithms and data. The design and implementation of the library has been strongly influenced by object-oriented principles.

There are two different styles of file formats available in VTK. The simplest are the legacy format. This file formats consist of five basic parts.

1. The first part is the file version and identifier. This part contains the single line: # vtk DataFile Version x.x.

2. The second part is the header. The header consists of a character string terminated by end-of-line character `\n`. The header can be used to describe the data and include any other pertinent information.

3. The next part is the file format. The file format describes the type of file, either ASCII or binary. On this line the single word ASCII or BINARY must appear.

4. The fourth part is the dataset structure. The geometry part describes the geometry and topology of the dataset. This part begins with a line containing the keyword DATASET followed by a keyword describing the type of dataset.

Then, depending on the type of dataset, other keyword/data combinations define the actual data.

5. The final part describes the dataset attributes. This part begins with the keywords POINT_DATA or CELL_DATA, followed by an integer number specifying the number of points or cells, respectively. (It doesn’t matter whether

POINT_DATA or CELL_DATA comes first.) Other keyword/data combinations then define the actual dataset attribute values (i.e., scalars, vectors, tensors, normals, texture coordinates, or field data).

According to the describing of the legacy VTK file formats, comparing our .vtk output files, we found out that the files we got were exactly the legacy formats.

The first part of our file shows that the version of our files is *3.0*. The header of our file is *vtk output*. The second part of our file tells us that the file format is *BINARY*. A binary file is computer-readable but not human-readable. It is encoded in binary form (which uses two symbols 0 and 1) for computer storage and processing purpose. The fourth part is the data structure. The dataset format of our file is *POLYDATA* (polygonal dataset). This polygonal dataset consists of arbitrary combinations of surface graphics vertices, and triangle strips. Polygonal data is defined by the *POINTS* and *TRIANGLE_STRIPS* sections.

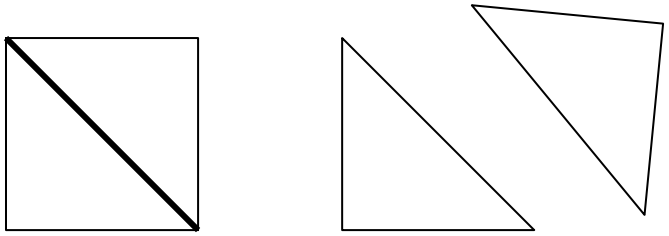
2.5.1.2 Points

As our dataset are triangle mesh produced by marching cubes algorithm (chapter 2.4.1), the point dataset must be 3D structured. The parameter *60968 float* shown in Figure 2.12 tells us the number and type of the points.

2.5.1.3 Triangle strips

As we know, the surface data is represented as a triangle mesh. A triangle mesh is a set of faces (triangles) in $\mathbf{R}^3$, such that the intersection between any pair of faces is either a common edge or nothing (Figure 2.13).

Triangle meshes consist of a list of vertices and a set of outlines. Each vertex contains a position and other additional data, such as diffuse and specular colors, shading normal, texture coordinates, etc. Each triangle outline has a list of integer indices. Each index is a number from 0 to n-1, where n is the number of vertices, and so points to a vertex in the list.



A common edge Nothing

Figure 2.13: Intersection between any pair of faces.

The triangle based mesh is packed by triangle strips as shown in Figure 2.14. [19][20]A triangle strip is a set of triangles where each triangle shares two vertices with the preceding triangle. The first three indices define a triangle and then each additional index defines another triangle by using the two preceding indices. E.g. in Figure 2.14 shows triangles 1, 2, 3, 4 with vertices A, B, C, D, E, F. By using triangle trip, the triangles can be represented as ABCDEF.

Triangle trip is a very effective way to store data. By this way to connecting triangles make it possible for fast rendering and efficient memory usage. As the example upon, if without using triangle strips, the triangles in Figure 2.14 would have to be stored four separate triangles: ABC, CBD, CDE, and EDF.

In this file shown in Figure 2.14, the two parameters *25401 197365* tells us the number of triangle trips and the total number of integer values required to represent the triangle trips.

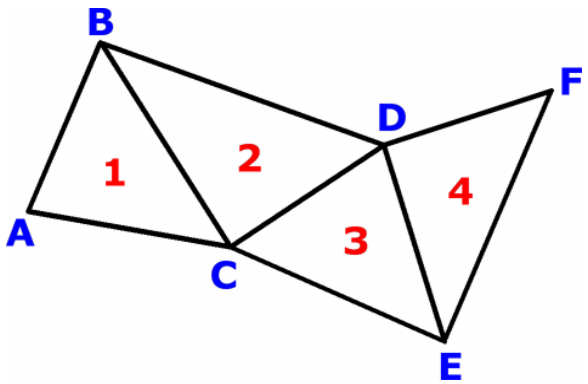


Figure 2.14: Triangle trip. [19]

2.5.1.4 Face normals

From Figure 2.12 we can see that the final part of the file is the dataset attribute values, normals. Before interpret what the normals mean. We have to employee 'Lighting' and 'Shading'. [21]Lighting is the term that is used to designate the interaction between material and light sources. Shading is the process of performing lighting computations and determining pixels' colors from them. There are three main types of shading: flat, Gouraud, and Phong. These correspond to computing the light per polygon, per vertex, and per pixel. In flat shading, a color is computed for a triangle and the triangle is filled with that color. In Gouraud shading, the lighting at each vertex of a triangle is determined, and these lighting samples are interpolated over the surface of the triangle. In Phong shading, the shading normals stored at the vertices are used to interpolate the shading normal at each pixel in the triangle. This normal is then used to compute the lighting's effect on that pixel. See Figure 2.15.

Therefore to calculate the light intensity in relation to a triangle, we have to first find the normal of that triangle. Normal is a normalized vector. It can be obtained in 3D space by using cross product of two of the triangle edges.

From the file shown in Figure 2.12, we can find out that the number of points and normals are identical, this indicates us that the 3D graphic uses Gouraud shading.

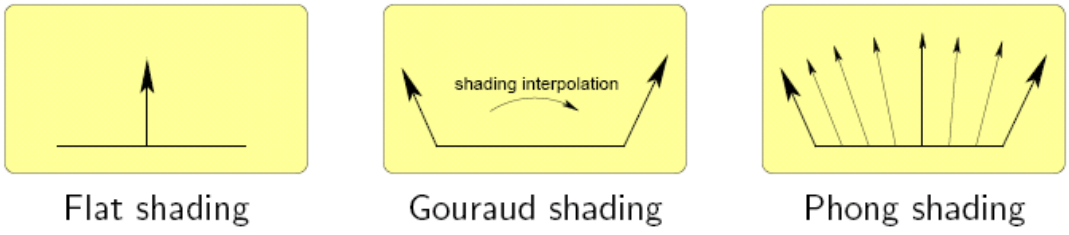


Figure 2.15: Shading. [22]

2.5.2 Reading file

VTK applies a class called vtkPolyDataReader which can help us reading such legacy formatted .vtk files. The process of file reading is as follow:

1.Read points

number of points = vtkPolyDataReader::GetOutput()::GetNubmerOfPoints()

for each **point** in **number of points**:

    point = vtkPolyDataReader:: GetOutput()::GetPoint()

2. Triangle-strip

number of cells = vtkPolyDataReader:: GetOutput()::GetNumberOfCells()

for each **cell** in number of **cells**:

    triangle_strip = point::GetId()

3. Read normals

number of normals = vtkPolyDataReader:: GetOutput()::GetPointData::GetNormals:: GetNumberOfTuples()

for each **normal** in number of **normals**:

    normal = vtkPolyDataReader:: GetOutput()::GetPointData::GetNormals::GetTuple()

The 3D graphic can not be seen after reading the file. Before an object is rendered, they must be placed within a screen.

2.5.3 Rendering

The points we extracted from the files are in world space. In world space the models are placed in the actual 3D world. To place an object on the screen, the world space coordinate system has to be transformed into view space, the space which is similar to world space and in which the origin is at the viewer or camera. The view direction (where the viewer is looking) defines the positive Z axis. Now the camera is now at the origin of the coordinate system, looking straight down the z-axis into the scene. Projection transformation then converts vertices from view spaces to projection space. In projection space, X and Y coordinates of a vertex are obtained from the X/Z and Y/Z ratios of this vertex in 3D space.

“Coin 3D” is graphic tool which can transform the object from world space to screen space.

Coin3D was created in 1995 and is a high-level 3D graphics toolkit for developing cross-platform real-time 3D visualization and visual simulation software. [24]Coin3D is portable over a wide range of platforms: any UNIX / Linux / *BSD platform, all Microsoft Windows operating systems, and Mac OS X. Coin3D is built on OpenGL and uses scene graph data structures to render 3D graphics in real-time. Basic import, rendering, and interaction with a 3D object can be implemented in just a few lines of code, and programmer efficiency is greatly increased compared with programming directly with OpenGL. OpenGL code and Coin3D code can co-exist in the same application, which makes gradual migration from OpenGL to Coin3D possible. Coin3D is fully compatible with SGI Open Inventor 2.1, the de facto standard for 3D visualization and visual simulation software in the scientific and engineering community. Additional features in Coin3D include VRML97 support, 3D sound, 3D textures, and parallel rendering on multiple processors. The core 3D rendering library Coin uses scene graph data structures to store and render graphics. Its data-driven design (redraws are only scheduled if the scene data changes) makes Coin3D well suited for user interface-based applications (e.g. scientific and engineering visualization applications), since it does not claim excessive CPU resources. Being only a 3D rendering library, Coin needs a user interface binding to be able to open windows, handle user input etc. GUI Bindings are available for the native Microsoft Windows GUI (SoWin), Trolltech's QT (SoQt), Xt/Motif on X Windows (SoXt), and the native Mac OS X GUI (Sc21). Coin3D comes with several file import/export libraries: simage for loading and saving images, sound and video files; Dime for loading, manipulating, and saving DXF files; Profit for loading, manipulating, and saving MultiGen Open Flight 3D model files.

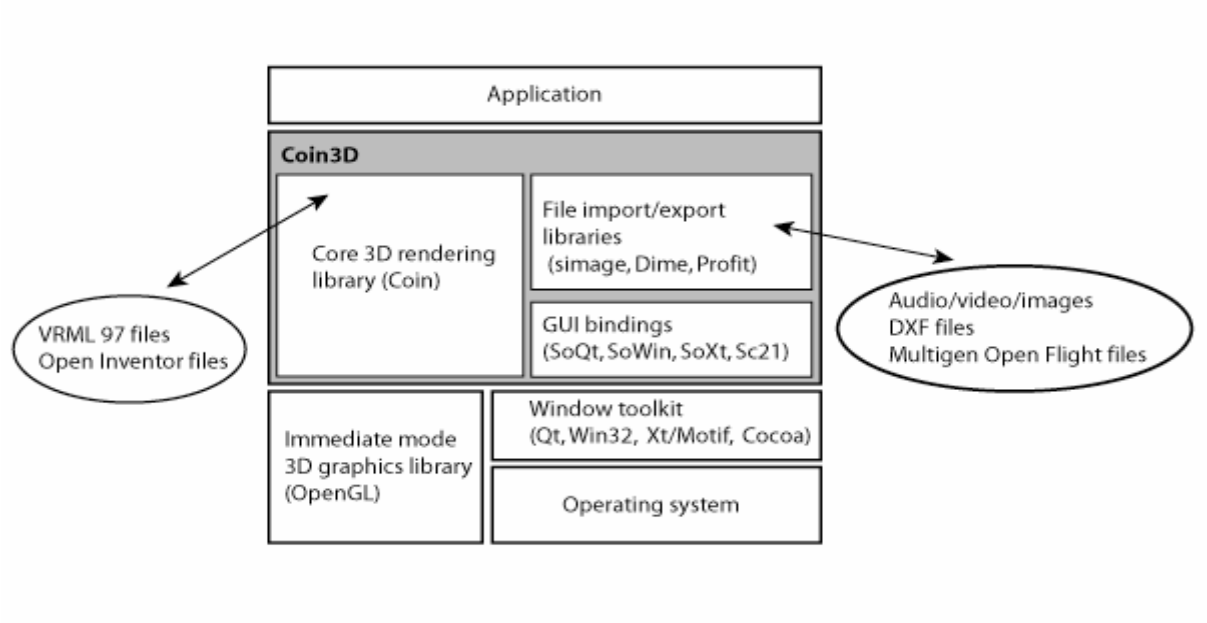


Figure 2.16: Coin3D libraries.

The node is the basic building block used to create three-dimensional scene database in Coin. Each node holds a piece of information such as a surface material, shape description, geometric transformation, light or camera. All 3D shapes, attributes, cameras, and lights sources presented in a scene are represented as nodes.

An ordered collection of nodes is referred to as a scene graph (Figure 2.17). Each node represents a geometry, property or grouping object. The scene graph is stored in the Coin scene database. A hierarchical scene can be created by adding nodes as children of grouping nodes, resulting in a directed acyclic graph. Coin takes care of storing and managing the scene graph in the database. After constructing a scene graph, we can apply a number of operations and actions to it, e.g. rendering, picking, searching, computing a bounding box, and writing to a file.

In our study, we want to show the geometric liver models one by one. With the combination of the object's speed and the time interval, we can get a coarse

view of the liver's motion during respiratory cycle. And we also want to use keyboard to control the object, so that we can have a view of the inside of liver.
To get an animation effect, we used the node call *SoBlinker*, which cycles among its children by changing the value of the *whichChild* field. And by using field *speed* can we control how many cycles per second. Figure 2.18 indicates the path for the node for showing our liver models on the scene.

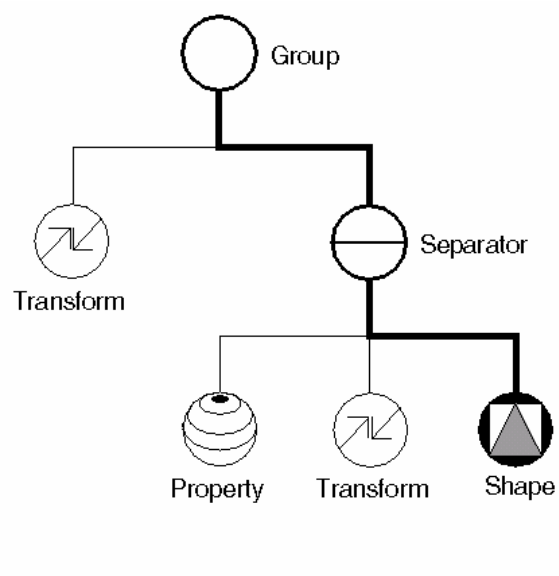


Figure 2.17: Scene graph.

The process of visualize our geomtric liver models $m_0, m_1, \ldots, m_5$ is as follow:
1. Read the surface file one by one by using the method presented in chapter 2.5.2. Then save the read data in a data structure, which contains attributes points, triangle trips and normals. Then put the data structure into a list structure (Figure 2.19).
2. As shown in Figure 2.18, the node *SoCoordinate3* sets the current coordinates of points in the rendering state; the node *SoIndexedTriangleStrip* constructs triangles strips out of the vertices located at the current coordinates; the node *SoNormal* sets the current surface normals in the rendering state to the specified vectors. By applying our data from step 1 to the corresponding nodes, we can get the liver models appear on the scene. By setting a value in the *speed* field of the *SoBlinker* node, we can get an animated liver.

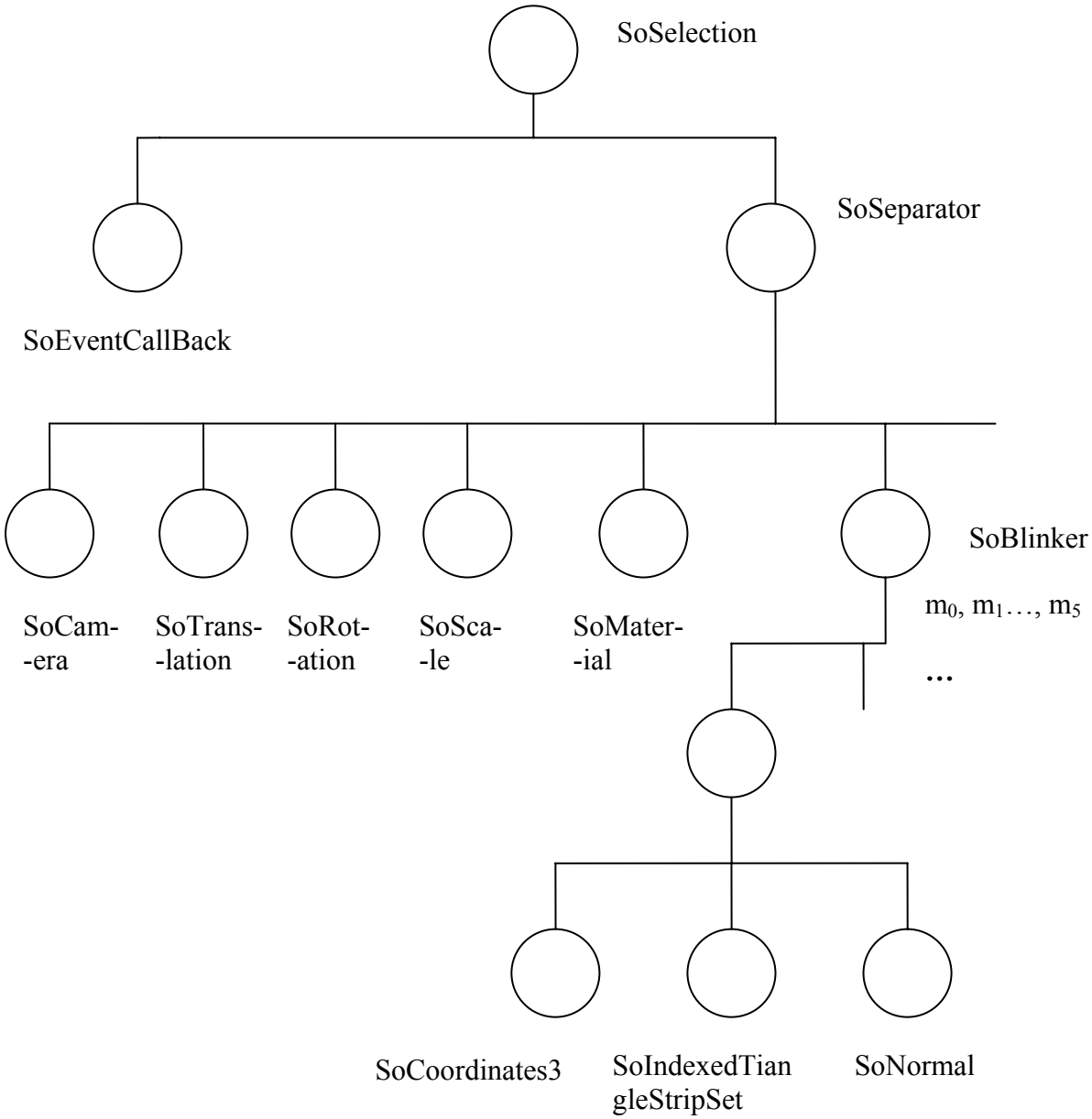


Figure 2.18: Scene graph showing geometric liver models.

3. To take a look inside the liver, we can use the *SoTranslation* node, to move the scene object towards the camera. The effect is like we looking inside the liver. By using node *SoRotate*, we can rotate the object. The effect is like we rotate the camera inside the object. If we want to see a specified place of the object, we can use the *SoScale* node, to enlarge the object.

4. We can use the *SoEventCallBack* node to make our own keyboard event and trigger the translation, rotate and scale event by using our keyboard. The object on the scene can then be controlled by the specified keys.

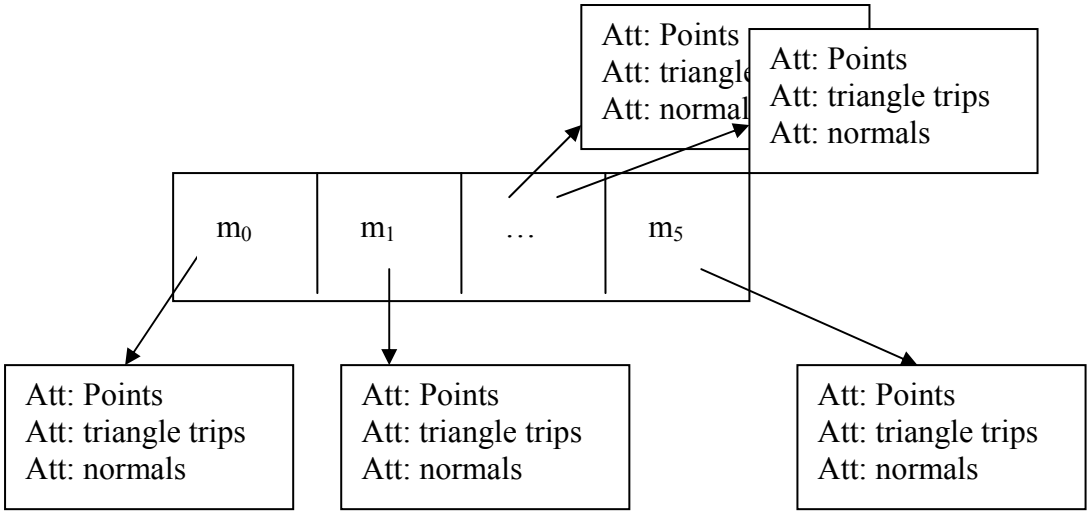


Figure 2.19: Data structure.

5. The animation effect is very jumping because we only have six set of data, and there is no interpolation between them. By applying a node call *SoGetBoundingBoxAction* (see also chapter 2.6.3), we can get the center points of the objects. We take these center points as control points, to use either linear interpolation or cubic B-spline interpolation to improve the animation effect. To let the animation express the process of the whole respiratory cycle, we add the model m_0 after m_5, to be the state "end of expiration".

The linear interpolation is shown in Equation 2.5.1, here c_1 and c_2 are two center points of two liver models from two adjacent respiratory points. The result of the computation is the number c which must lie between c_1 and c_2.

$$c = (1 - \lambda)\, c_1 + \lambda\, c_2 \qquad \textbf{(E.q. 2.5.1)}$$

The cubic B-spline interpolation is: Let d be a positive integer and let the d+1 points $(c^i_{j=i-d})$ be given together with the 2d knots $t = (t_j)^{i+d}_{j=i-d+1}$. The point $p_{i,d}$ of

degree d is determined by the following computation. First set $p_{j,0}(t) = c_j$ for $j = i-d, i-d+1, ...i$ and use Equation 2.5.2 to compute. Here the value of d is 3.

$$p_{j,r}(t) = \frac{t_{j+d-r+1} - t}{t_{j+d-r+1} - t_j} p_{j-1,r-1}(t) + \frac{t - t_j}{t_{j+d-r+1} - t_j} p_{j,r-1}(t) \qquad \textbf{(E.q.2.5.2)}$$

$$For \ \ j = i-d+r, ..., i \ \ and \ \ r = 1,2,...,d.$$

Since we have now seven control points, the knot sequences are: (0,0,0,0,1,2,3,4,4,4,4).

2.5.4 Visualization by "3DSlicer"

"3D Slicer" is a tool combines capability for segmentation, visualization and registration. Once a 3D structured model is generated, it will be displayed on the main 3D screen in 3DSlicer. The model can be rotate in 360 degrees. By applying the "Clipping" module in "3D Slicer", we can cut away sections of a model at one or more selected slicers. [25] The geometric model is removed from one side of each selected slice, revealing the image of each selected slice and the rest of the visible model (Figure 2.20).

By applying a visualize module, "virtual endoscope". It provides a technique for interactive exploration of the inner surface of the 3D model while being able to track the position of the virtual endoscope relative to the 3D model (Figure 2.21).

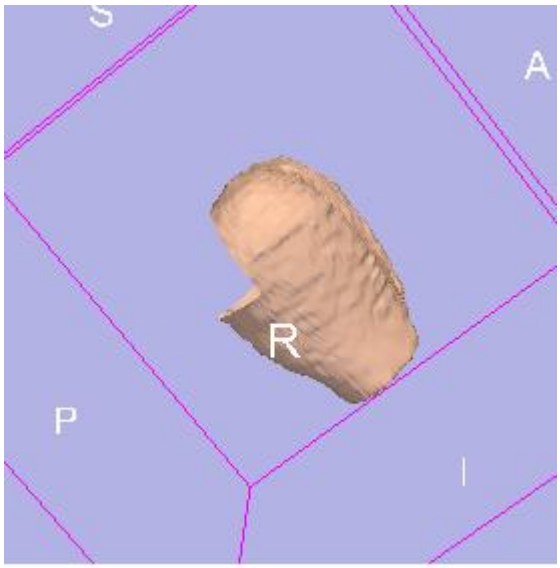


Figure 2.20: Clipped model.

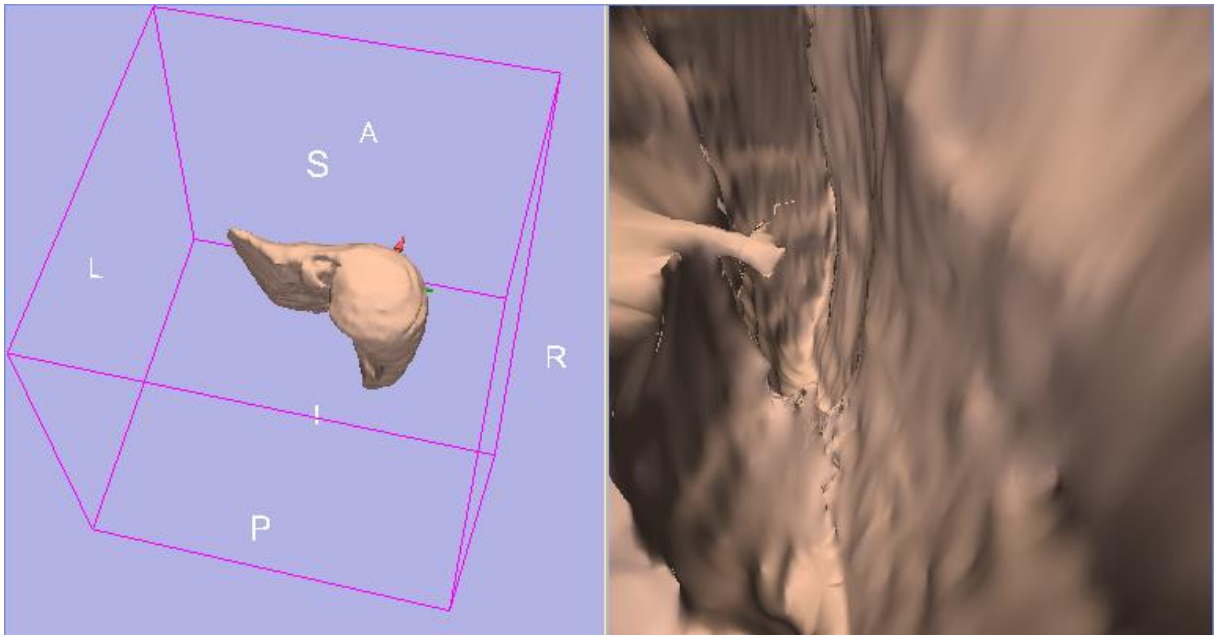


Figure 2.21: Virtual Endoscope.

2.6 Rigid registration

The transformations of the liver during the respiratory cycle include both overall motions and local motions. The overall motion is called global motion. It can be describes as rigid transformation. Rigid transformation is any transformation which leaves the shape and size of object unchanged. We can use rigid registration to catch the global motion of the liver. Rigid registration of 3D images is the process of aligning different images into a common coordinate system by applying certain rotations and translations [26]. The process of rigid registration is to first define a reference data set and a data set which is going to move, then rigidly aligning the moving data set so that the shape of objects is as closed to each other as possible while keeping the single shapes unchanged. In the study here, the data sets are the segmented volume data – labelmaps (chapter 2.3.2).

The motivation of rigid registration in this study is to bring multiple image data sets into spatial alignment and detect the movement between two liver volumes.

2.6.1 Rigid transformation

The global motion can be described as rigid transformation, which is any transformation that leaves the shape and size of object unchanged. i.e. Suppose an object is moved from one position to another. In this process, any point P of the object will be moved to another point P'. This is said that the points of the object are transformed into other points. A transformation is said to be rigid if it preserves relative distances, that is to say, if P and Q are transformed to P' and Q' then the distance from P to Q is the same as that from P' to Q'.

Rigid transformation includes two basic transformations: Translation and rotation. In order to not missing the direction of the translation, we are going to use homogeneous notation to represent our rigid transformations.

2.6.1.1 Homogeneous notation

Since a point describes a location in space, it is not possible to express a translation as a multiplication by using 3×3 matrices (see chapter 2.6.4). However, homogeneous notation includes the ability to perform translation on points [21]. It augments 3×3 matrices to the size 4×4, and three-dimensional points get one more element. So a homogeneous vector is $P=(p_x,p_y,p_z,p_w)$.

Equation 2.6.1 shows how a 3×3 matrix, M, is augmented into the homogeneous form.

$$M_{4\times4}=\begin{pmatrix} m_{00} & m_{01} & m_{02} & 0 \\ m_{10} & m_{11} & m_{12} & 0 \\ m_{20} & m_{21} & m_{22} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \textbf{(E.q.2.6.1)}$$

2.6.1.2 Translation

A change from one location to another is represented by a translation matrix, *A*. This matrix translates an entity by a vector $t=(t_x,t_y,t_z)$.

$$T(t) = T(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \textbf{(E.q.2.6.2)}$$

A point $p = (p_x, p_y, p_z, 1)$ with $T(t)$ yields a new point $p' = (p_x + t_x, p_y + t_y, p_z + t_z, 1)$, which is a translation.

2.6.1.3 Rotation

Rotation matrices is represented as $R_x(\phi), R_y(\phi),$ and $R_z(\phi)$, which rotate an entity ϕ radians around the x-, y- and z-axes respectively. They are given by following equations:

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \textbf{(E.q.2.6.3)}$$

$$R_y(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \textbf{(E.q.2.6.4)}$$

$$R_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \textbf{(E.q.2.6.5)}$$

A rigid transformation H is an image coordinate transformation composed of a translation matrix, T and a rotation matrix, R. Thus H has the appearance of the matrix in Equation 2.6.6.

$$H = TR = \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \textbf{(E.q.2.6.6)}$$

2.6.2 Rigid registration by “3D Slicer”

The “3D Slicer” supports rigid, manual registration, auto registration and semi-auto registration. Manual registration allowing the user to specify which volume to move, and then translate and rotate that data set by clicking and dragging on the images in the 2D windows. Auto registration allowing the user to specify which volume is reference volume and which volume to move, then translate and rotate the data set automatically. Semi-auto registration is an interactive registration method between 3D Slicer and user, which is a combination of manual registration and auto registration.

The processing pipeline of rigid registration using 3D Slicer is shown as Figure 2.22.

2.6.2.1 Loading data and visualize misalignment

The data which is used here is the labelmaps l_0, l_1, ...,l_5 generated by segmentation. In order to catch rigid transformations between two adjacent points in the respiratory cycle, we perform six registrations to our labelmaps. Table 2.6.1 specifies the reference volume and the moving volume.

After loading the data sets, the reference volume is set to background and the registered volume is set to foreground.

The labelmap which is filled with pink color is the registered volume. The labelmap which is filled with background color is the reference volume.

Then observe the misalignment between the two volumes. Figure 2.23 shows clearly misalignment between the two volumes.

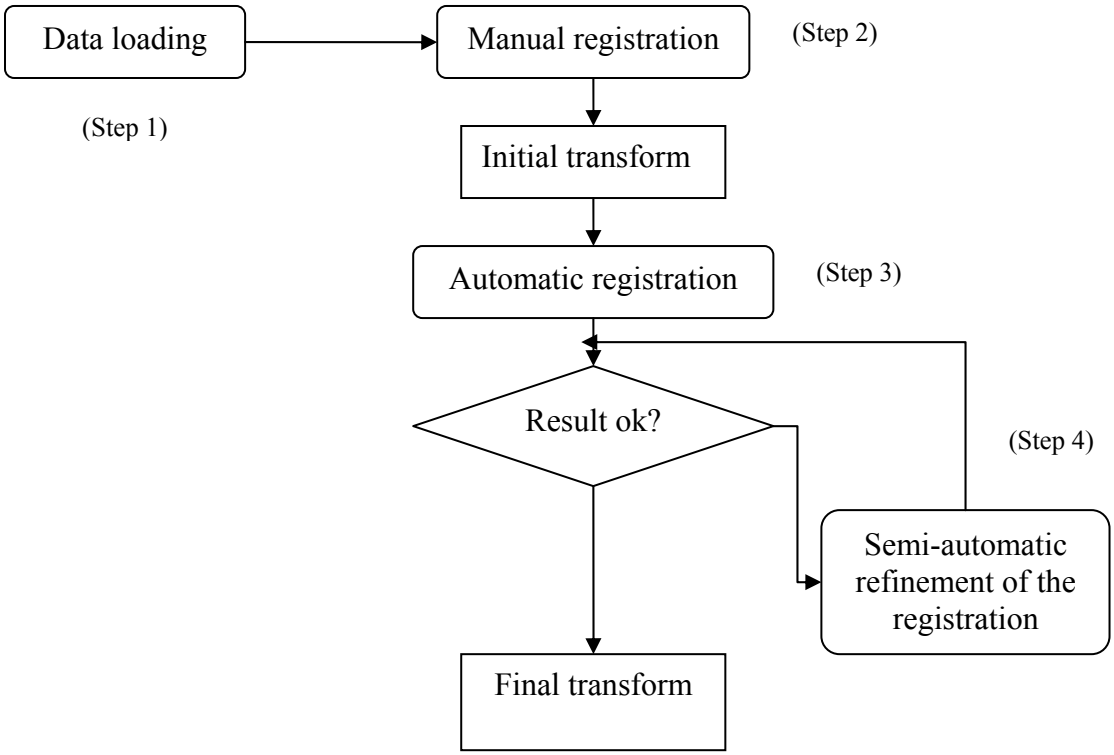


Figure 2.22: The pipeline of rigid registration.

Registered volume	Reference volume
l_1	l_0
l_2	l_1
l_3	l_2
l_4	l_3
l_5	l_4
l_6	l_5

Table 2.6.1: Registered volumes vs. reference volumes.

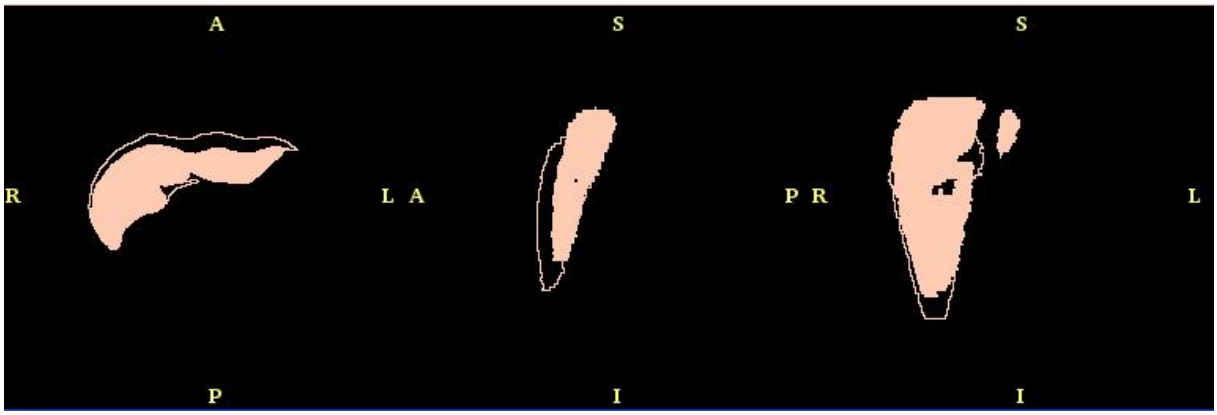


Figure 2.23: Misalignment between two volumes.

2.6.2.2 Manual registration

If misalignment is found between two volumes, a rigid transform has to be applied to the registered volume. A transformation can be added by manually moving the volume. 3D Slicer displays the value of the applied manual translation in Figure 2.24. And Figure 2.25 shows the alignment of the two volumes in Figure 2.23 after manual registration. If a clear misalignment still exits, an automatic registration can be applied.

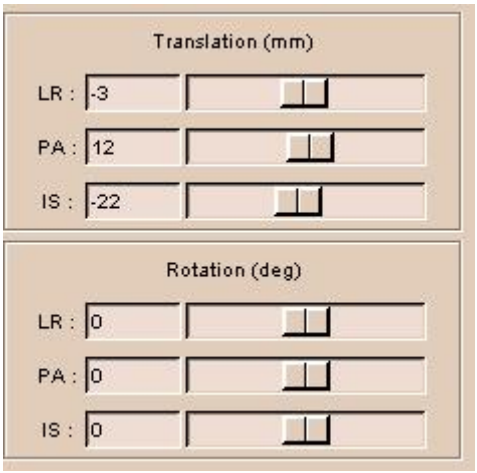


Figure 2.24: Result from manual registration.

2.6.2.3 Automatic registration

By selecting the panel Auto in the module Alignments in 3D Slicer, an automatic registration can be started. This auto registration translates and rotates volume automatically. The result is shown as Figure 2.26. If the result is still unsatisfying, 3D Slicer applies a semi-automatic registration mode.

2.6.2.4 Semi-automatic registration

In the semi-automatic mode, the transformation can be interactively manipulated during the registration process. And user can stop manually this mode to obtain the final value of the registration matrix. Figure 2.27 shows the result after semi-automatic registration.

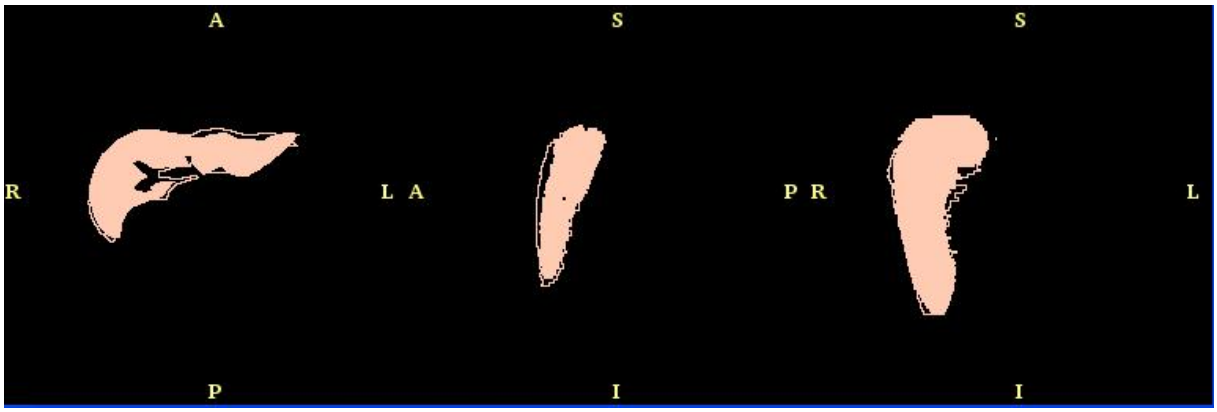


Figure 2.25: Alignment between two volumes after manual registration.

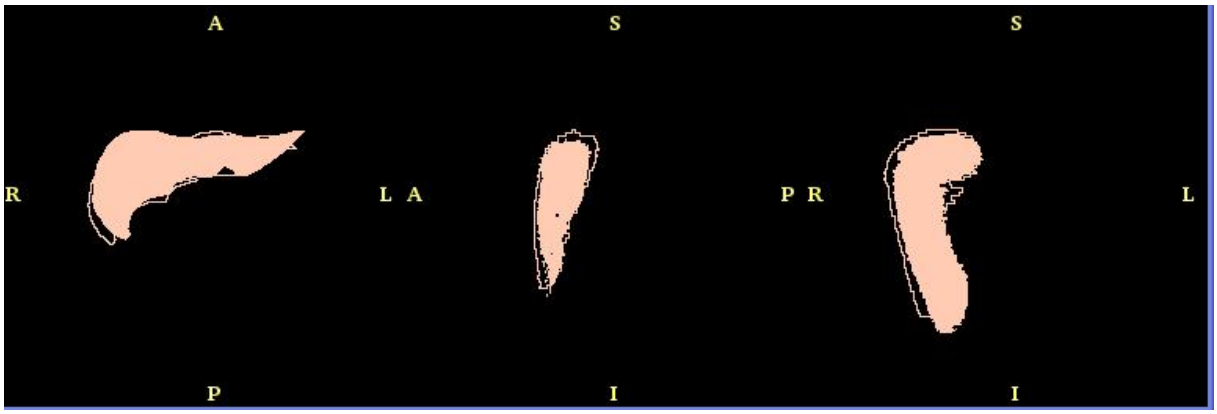


Figure 2.26: Alignment between two volumes after automatic registration.

2.6.3 Bounding box

"Coin3D" applies the bounding-box action, which computes a 3D bounding box that encloses the shapes in a sub-graph under a node or defined by a path. This action also computes the center point of these shapes. By applying this function on the liver models, we can obtain a bounding box and the center point for the objects in world coordinates. By comparing the center point one by one according to Table 2.6.1, we can also get six transformations. This can help us check the validation of the output results from rigid-registration.

2.6.4 Concatenation of transforms

If we want to get the transformations between the two sampling points in the respiratory cycle which are not adjacent, we can perform rigid registration

between the two lablemaps which were got from the sampling points. The more information of transformations we want to know, the more rigid registrations we have to do. This can be a very large job. We have an easy way to resolve the problem:

Assume we have point *p* which transformed to *p'* by *M*, We have:

$$P' = Mp$$

We can then transform *p'* by *N* to get *p''*

$$P'' = Np'$$

Which is equivalent to

$$P'' = NMp$$

That is to say, any number of transforms $M_1 \ldots M_n$ can be concatenated into one single transform *M*.

$$M = M_n M_{n-1} \ldots M_2 M_1 \qquad \textbf{(E.q.2.6.7)}$$

By rigid registration, we got six set of data, and every set contains two transforms *T*, *R*, one for translation and the other for rotation. We can concatenation the two transforms, *TR* and we use *H* to represent it. The transforms is shown in Figure 2.28. If we want to know the transforms between to points, we can just concatenate all the transforms between them as shown in equation 2.6.7.

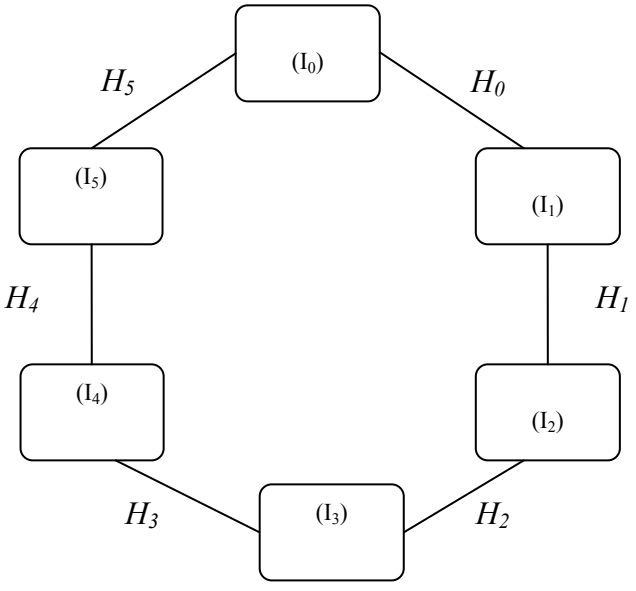


Figure 2.28: The global transformations.

2.6.5 Volume calculation

In addition to the rotation and translation, we also interested in whether the volume of liver changes during respiratory cycle. We can computer the volume of liver by the labelmaps. By calculating the summation of all the voxels in one labelmap set, we can get the volume of the liver belongs to that labelmap set. This job seems to be very time-consuming. Since we have 3D mesh models m_0, m_1, … , m_5. We can calculate the volume of liver by the 3D geometric model. The process of such volume calculation is as follow [27]:

In the calculation, the elementary calculation unit is a tetrahedron, which is a polyhedron composed of four triangular faces, three of which meet at each vertex. For each triangle, we connect each of its vertices with the origin and form a tetrahedron, as shown in Figure 2.29. We define the signed volume for each elementary tetrahedron as: The magnitude of its value is the volume of the tetrahedron, and the sign of the value is determined by checking if the origin is at the same side as the normal with respect to the triangle. In Figure 2.29, triangle *ACB* has a normal N_{ACB}. The volume of tetrahedron *OACB* is:

$$\left|V_{OACB}\right| = \left|\frac{1}{6}(-x_3y_2z_1 + x_2y_3z_1 + x_3y_1z_2 - x_1y_3z_2 - x_2y_1z_3 + x_1y_2z_3)\right|$$

As the origin *O* is at the opposite side of N_{ACB}, the sign of this tetrahedron is positive. The sign can also be calculated by inner product $\overrightarrow{OA}\cdot N_{ACB}$. In real implementation, again we only need to compute:

$$V_i^{'} = \frac{1}{6}(-x_{i3}y_{i2}z_{i1} + x_{i2}y_{i3}z_{i1} + x_{i3}y_{i1}z_{i2} - x_{i1}y_{i3}z_{i2} - x_{i2}y_{i1}z_{i3} + x_{i1}y_{i2}z_{i3})$$

$$V_{total}^{'} = \sum_i V_i^{'}$$

Where *i* stands for the index of triangles or elementary tetrahedrons. $(x_{i1}, y_{i1}, z_{i1}), (x_{i2}, y_{i2}, z_{i2}), (x_{i3}, y_{i3}, z_{i3})$ are coordinates of the vertices of triangle *i* and they are ordered so that the normal of triangle *i* is consistent with others.

Volume of a 3D mesh model is always positive. The final result can be achieved by take the absolute value V'_{total}.

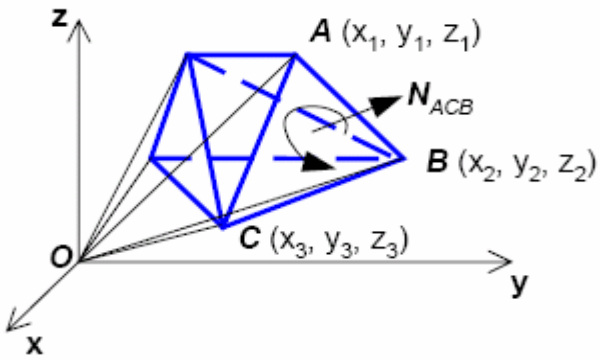


Figure 2.29: Calculation of 3D volume.

It can be easier to just use the module applied by "3D Slicer", called "Measure". This module applies us solution for measure both volume and area of a 3D mesh volume. It takes an input of a mesh volume, and the result is calculated just by click the "Measure" button.

2.7 Non-rigid registration

As mentioned before, the liver presents some local deformations in addition to global deformation through the breathing cycle. And these complex deformations during respiration cannot be measured by a simple rigid registration.

To find these local deformations, the simplest and idealist method is to first apply the global deformation to the vertex coordinates of the geometric models of the liver and then compare the position of each node of the geometric models. E.g. as in Figure 2.28, add H_0 to every point in the geometric model m_0 derived from I_0, then compare the transformed position of each point in m_0 with the correspond point in the geometric model m_1 derived from I_1. By this method we

can find out those vertices that have most position difference. And by using 3D visualize software, such as Coin3D, we can directly find out those areas in the liver have the most local deformations during respiration. But the precondition of this method is that the number of vertices of all models should be identical. According to our study, the model was generated after B-spline segmentation, and during segmentation we cannot guarantee the geometric models we get have the identical number of vertices.

Another way is using non-rigid registration to align the data sets. Non-rigid registration is the general term for the alignment of data sets that are mismatched in a nonlinear or non-uniform manner. [30]A method to do non-rigid registration is using free-form deformation. Free-form deformation (FFD) is a technique for manipulating any shape in a free-form manner.

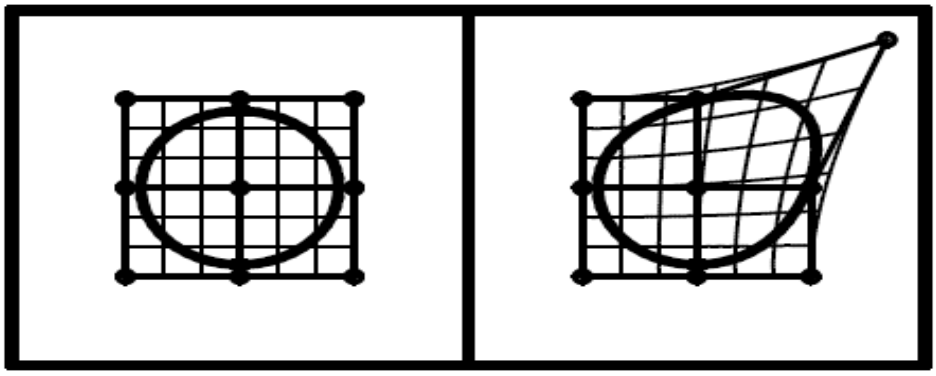


Figure 2.30: Free-form deformation.

The basic idea of FFD is to deform an object by manipulating an underlying mesh of control points. In free-form deformation, a localized coordinate grid, in a standard configuration, is superimposed over an object. For each vertex of the object, coordinates relative to this local grid are determined that register the vertex to the grid. The grid is then manipulated by the user. Using its relative coordinates, each vertex is then mapped back into the modified grid, which relocates it in global space (Figure 2.30). A cubic interpolation is typically used with FFD. The resulting deformation controls the shape of the 3-D object and produces a smooth and C^2 continuous transformation [31].

2.7.1. Free-form deformation based on B-splines.

The free –form deformation based on B-splines has been previously applied by D. Rueckert[32] to describe the local motion of breast. And was later be applied by Torsten Rohlfing[3] to computer the deformation field between liver images. This method has been convinced as a powerful tool for modeling 3-D deformable objects. The procedure of this method is as follow:

The FFD method deforms an object by first assigning to each of its points within the deformation lattice a set of local coordinates. Points of an object are located in a three-dimensional rectilinear grid. The local coordinate system is defined by an orthogonal set of three vectors (*S, T, U*). A point P is registered in the local coordinate system by determining its trilinear interpolate, as done in Equation 2.5.1.1, Equation 2.5.1.2, and Equation 2.5.1.3.

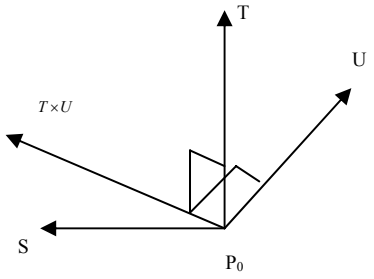


Figure 2.31: Initial local coordinate system for FFDs.

$$s = (T \times U) \bullet (P - P_0) / ((T \times U) \bullet S) \quad \textbf{(Eq.2.5.1.1)}$$

$$t = (U \times s) \bullet (P - P_0) / ((U \times S) \bullet T) \quad \textbf{(Eq.2.5.1.2)}$$

$$u = (S \times T) \bullet (P - P_0) / ((S \times T) \bullet U) \quad \textbf{(Eq.2.5.1.3)}$$

The equations above, the cross product of two vectors forms a third vector that is orthogonal to the first two. The denominator normalizes the value being computed. In equation 2.5.1.1, computes the projection of S onto $T \times U$ determines the distance within which points will map into the range $0 < s < 1$.

Given the local coordinates (*s, t, u*) of a point and the unmodified local coordinate grid, a point's position can be reconstructed in globe space by

moving in the direction of the local coordinated axes according to the local coordinate.

$$P = P_0 + s \bullet S + t \bullet T + u \bullet U \qquad \textbf{(Eq.2.5.1.4)}$$

To facilitate the modification of the local coordinate system, a grid of control points is created in the parallelepiped defined by the *S*, *T*, *U* axes. All object points within this parallelepiped are assigned local coordinates through a mapping applied to their xyz-coordinate. For any point P_{stu} in the local coordinates, the deformation at this coordinate is computed from the position of the surrounding control points.

To define the control points, let *i*, *j* and *k* denote the index of control point cell which containing P_{stu}, and δ_x, δ_y and δ_z denote the spacing of control points. *i*, *j* and *k* can be computed as:

$$i = \left\lfloor \frac{s}{\delta_x} \right\rfloor - 1,\ j = \left\lfloor \frac{t}{\delta_y} \right\rfloor - 1,\ k = \left\lfloor \frac{u}{\delta_z} \right\rfloor - 1 \qquad \textbf{(E.q.2.5.1.5)}$$

The relative position of P_{stu} in the local three dimensions within the cell can be computed as:

$$s' = \frac{s}{\delta_x} - (i+1),\ t' = \frac{t}{\delta_y} - (j+1),\ u' = \frac{u}{\delta_z} - (k+1) \quad \textbf{(E.q.2.5.1.6)}$$

The deformation at the coordinates is computed from the position of the surrounding 4×4×4 control points by using the basis function of the B-spline. The definition of basis B-spline functions is [33]:

$$B_{i,0}(t) = \begin{cases} 1(t_i \le t < t_{i+1}, t_i < t_{i+1}) \\ 0(otherwise) \end{cases}$$

$$B_{i,d}(t) = \frac{t - t_i}{t_{i+d} - t_i} B_{i,d-1}(t) + \frac{t_{i+d+1} - t}{t_{i+d+1} - t_{i+1}} B_{i+1,d-1}(t) \qquad \textbf{(E.q.2.5.1.7)}$$

A degree *d* spline curve *f* can be constructed from *n* control points $(c_i)_{i=1}^{n}$ and $n + d + 1$ knots $(t_i)_{i=1}^{n+d+1}$ and written as:

$$f = \sum_{i=1}^{n} c_i B_{i,d}$$

A cubic B-spline curve is when $d = 3$. According to Equation 2.5.1.7, the basis B-spline function can be written as:

$$B_0(t) = (1-t)^3/6, \quad \textbf{(E.q.2.5.1.8)}$$

$$B_1(t) = (3t^2 - 6t^2 + 4)/6, \quad \textbf{(E.q.2.5.1.9)}$$

$$B_2(t) = (-3t^3 + 3t^2 + 3t + 1)/6, \quad \textbf{(E.q.2.5.1.10)}$$

$$B_3(t) = t^3/6 \quad \textbf{(E.q.2.5.1.11)}$$

It is defined using four control points over the interval zero to one. A compound curve is generated from an arbitrary number of control points by constructing a curve segment from each four-tuple of adjacent control points: $(P_i \quad P_{i+1} \quad P_{i+2} \quad P_{i+3})$ for $i = 1,2,...,n-3$, where n is the total number of control points. Each section of the curve is generated by multiplying the same 4×4 matrix (Equation 2.5.1.12) by four adjacent control points with an interpolating parameter between zero and one.

$$x(t) = \frac{1}{6}\begin{bmatrix} P_i & P_{i+1} & P_{i+2} & P_{i+3} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \quad \textbf{(E.q.2.5.1.12)}$$

By using the basis function of cubic B-spline, for any location (x, y, z) in object, the deformation at these coordinates is computed from the position of the surrounding 4×4×4 control points:

$$\mathrm{T}(x,y,z) = \sum_{l=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} B_l(s) B_m(t) B_n(u) \phi_{i+l, j+m, k+n}$$

Where T is the non-rigid transformation which is defined on a uniformly spaced grid $\boldsymbol{\Phi}$ of control points $\Phi_{i,j,k}$, where $-1 < i < n_x - 1$, $-1 < j < n_y - 1$ and $-1 < k < n_z - 1$ which is overlaid on domain $\boldsymbol{\Omega}$, where $\boldsymbol{\Omega} = \{(x,y,z) \mid 0 < x < n_x,\ 0 < y < n_y,\ 0 < z < n_z\}$ is the domain in xyz-plan. Control points with i, j, or k equal to either 0 or $n_x - 3 (n_y - 3,\ n_z - 3)$ are located on the edge of the image data. For any location (x,y,z), the deformation at these coordinates is computed from the position of the

surrounding 4×4×4 control points. The *i, j* and *k* denotes the index of control point cell containing (*x,y,z*) and *s, t* and *u* are the relative position of (*x,y,z*) in the three dimensions. B_l, B_m and B_n are uniform cubic B-spline basis function defined as Equation 2.5.1.8, Equation 2.5.1.9, Equation 2.5.1.10 and Equation 2.5.1.11. Because B-spline are local controlled (see also chapter 2.3.2), it makes the computation more efficient for large number of control points. Changing of control points $\Phi_{i,j,k}$ affects the transformation only in the local neighborhood of that control point.

The process of non-rigid registration in our study is as follow:

1. Move all points to domain **Ω**. Implement "Bounding box" action on our geometric models and find out the maximum length, width and height. Then move the geometric models on domain **Ω**, where **Ω** = {(*x,y,z*) | *0<x<maxlength, 0<y<maxheight, 0<z<maxwidth*}.

2. Define grid. Define the spacing of control points. Here we use 20mm in all directions. Without loss of generality, we assume that grid $\boldsymbol{\Phi}$ is an (*maxlength/20+4*)(*maxheight/20+4*)(*maxwidth/20+4*) grid.

3. Register points in the local coordinate system. Every point P in the geometric models is registered in the local coordinate system, as done in Equation 2.5.1.1, Equation 2.5.1.2, and Equation 2.5.1.3.

4. Computer the deformation of every point P from the position of the surrounding 4×4×4 control points in each model. The indices of the control points that contain P are computed as Equation 2.5.1.5. And the relative position of P within that cell is defined as Equation 2.5.1.6.

5. For each point register the deformation on its control points. If the control points contain more than one point, just add all the deformations.

6. By comparing the average deformations on the control points model by model according to Table 2.6.1, we can the register the deformations of the corresponding points in the control points from one respiratory state to another.

2.7.2 “3DSlicer” – Deformable BSpline Transform

“3D Slicer” applies a module called “Deformable B-spline Transform”, which is equivalent to generation a deformation field where a deformation vector is assigned to every point in space. The deformation vectors are computed using B-spline interpolation from the deformation values of points located in a coarse grid, which is referred to as the B-spline grid. And this method is written in the itk file ‘BSplineDformableTransform.h’ [34]. This class encapsulates a deformable transform of points from one N-dimensional one space to another N-dimensional space. The deformation field is modeled using B-splines. A deformation is defined on a sparse regular grid of control points and is varied by defining a deformation of each control point. The deformation at any point is obtained by using a B-spline interpolation kernel. Each grid/control point has associated with it N deformation coefficients, representing the N directional components of the deformation.

Chapter 3

Result

In this chapter we will show the results from: 1. Globe movement of the liver; 2. Volume of the liver; 3. Visualization of the liver; 4. Local movement of the liver.

3.1 Global movement

By implementing "Boundingbox" action, we got the center points of the liver models from the six sampling points in respiratory cycle, as shown in Table 3.1 and Figure 3.1. In Figure 3.1 we added an extra point – center point of the liver from $I_{0,}$ after $I_{5,}$ to make the graph more comparable.

	I_0	I_1	I_2	I_3	I_4	I_5
R-L	**19.27**	**15.11**	**16.89**	**15.32**	**12.45**	**13.21**
A-P	**-4.23**	**5.08**	**10.04**	**13.91**	**2.16**	**0.71**
S-I	**15.18**	**-11.61**	**-14.67**	**-16.42**	**8.53**	**-4.58**

Table 3.1: The center points from the six liver models.

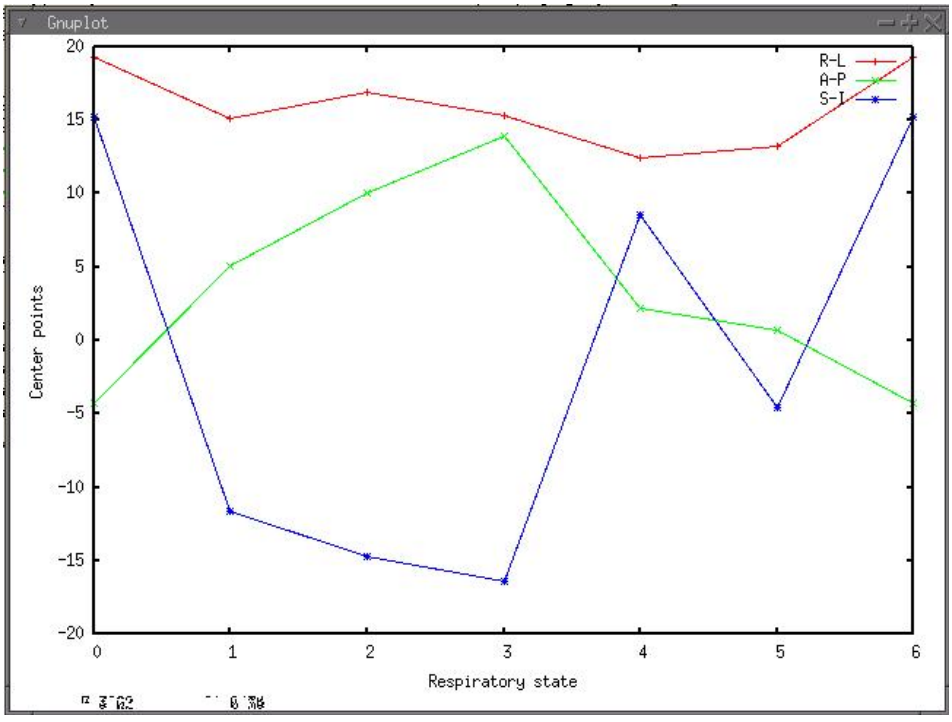


Figure 3.1: The center points from the six liver models. X-coordinate shows the sampling points. Y-coordinate shows the center points.

By calculating the movement of center points from two liver models of the adjacent respiratory points two and two, we can get the global movement of liver. Table 3.2 - Table 3.7 show the results of the global movement of liver by using "Bounding box" and "3D Slicer". The results from both methods have maximum 3mm difference. It means that our rigid registration data are valid.

	Bounding box	"3DSlicer"
A-P	**9.30**	**10.68**
S-I	**-26.79**	**-26.49**
R-L	**-4.16**	**-2.73**

Table 3.2: I_0-I_1

	Bounding box	"3DSlicer"
A-P	**4.96**	**5.82**
S-I	**-3.06**	**-2.36**
R-L	**1.78**	**3.00**

Table 3.3: I_1-I_2

	Bounding box	"3DSlicer"
A-P	**3.87**	**5.22**
S-I	**-1.75**	**-4.31**
R-L	**-1.57**	**-0.19**

Table 3.4: I_2-I_3

	Bounding box	"3DSlicer"
A-P	**-11.75**	**-11.43**
S-I	**24.95**	**28.75**
R-L	**-2.87**	**-2.10**

Table 3.5: I_3-I_4

	Bounding box	"3DSlicer"
A-P	**-1.45**	**0.78**
S-I	**-13.11**	**-13.56**
R-L	**0.76**	**-1.02**

Table 3.6: I_4-I_5

	Bounding box	"3DSlicer"
A-P	**-4.93**	**-3.70**
S-I	**19.76**	**18.69**
R-L	**6.06**	**3.74**

Table 3.7: I_5-I_0

Table 3.2-Table3.7: Rigid-registration using "Bounding box" vs. using "3D Slicer".

The volume of liver calculated from model m_0, m_1, … m_5 are show as Table 3.8 and Figure 3.2:

	m_0	m_1	m_2	m_3	m_4	m_5
Volume (ml)	1217.30	1375.15	1372.68	1418.17	1403.67	1372.2

Table 3.8: Volume.

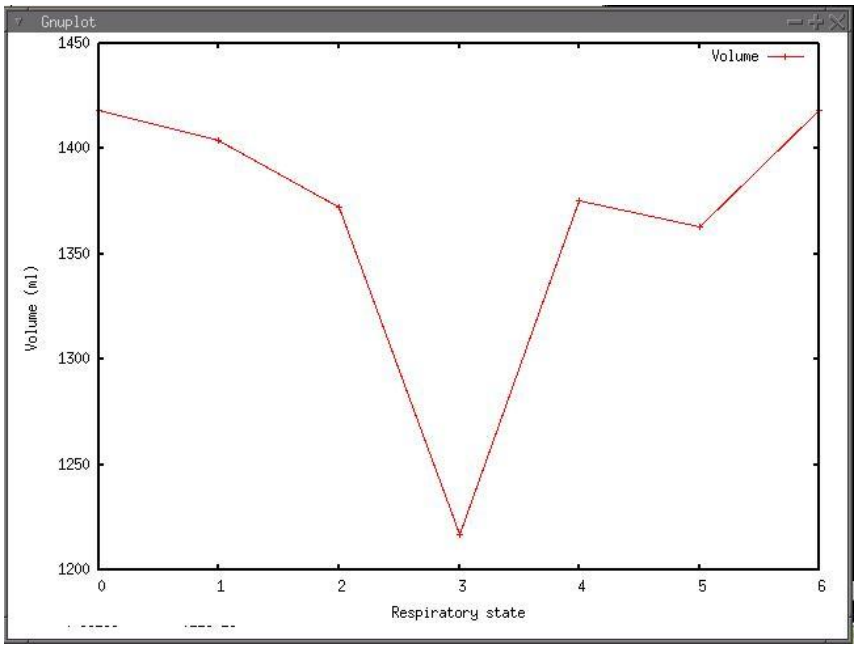


Figure 3.2: Volume.

3.2 Visualization

The 3D geometric models of liver are shown in Figure 3.3-3.8.

The path of the global motion of the liver interpolated by cubic B-spline is shown in Figure 3.9. Figure 3.9 a shows the path looking from the x-y plan; Figure 3.9 b shows the path looking from the x-z plan; Figure 3.9 c shows the path looking from the y-z plan.

In addition, by moving camera, we can see the inner structure of the liver. See Figure 3.10.

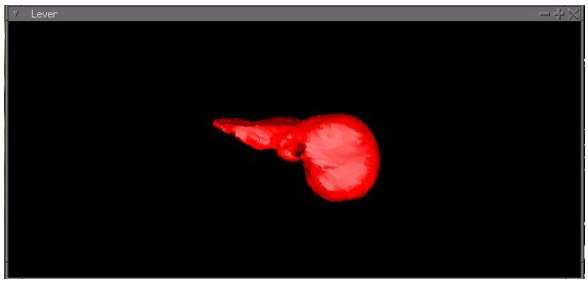


Figure 3.3: m_0

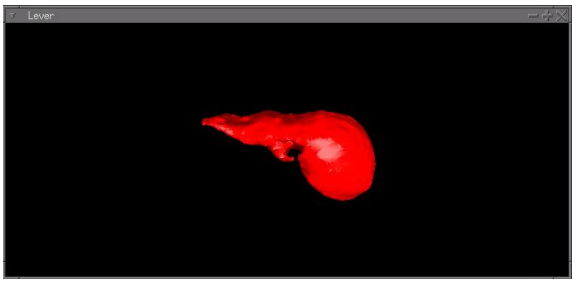


Figure 3.4: m_1

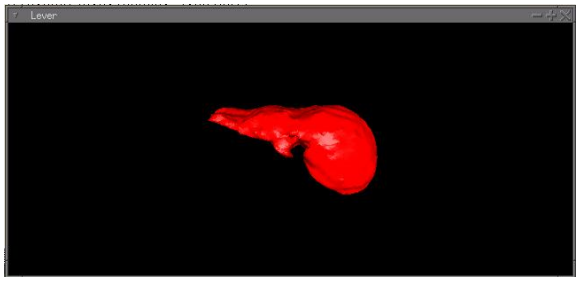


Figure 3.5: m_2

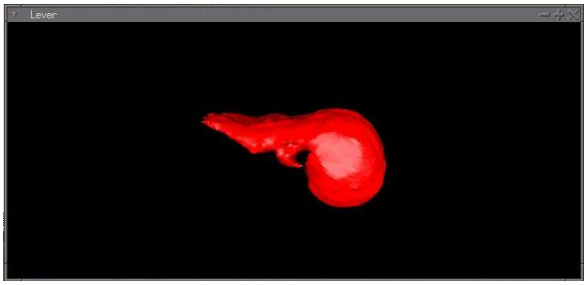


Figure 3.6: m_3

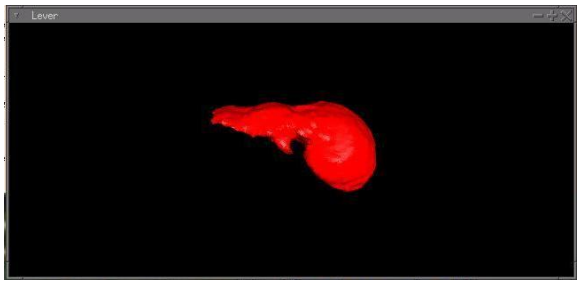


Figure 3.7: m_4

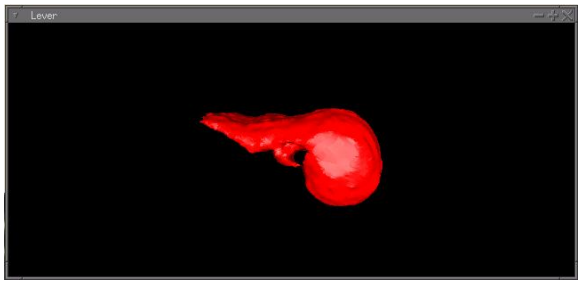
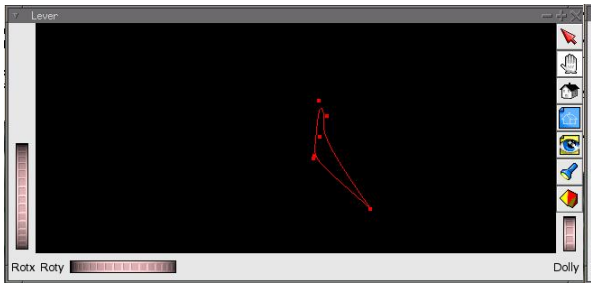
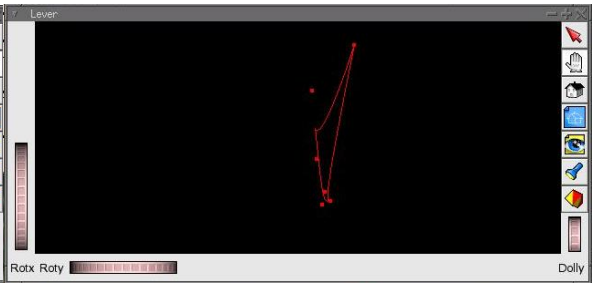


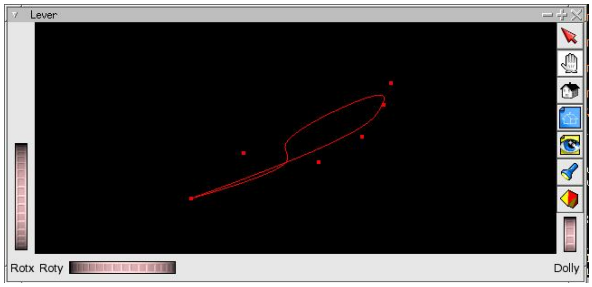
Figure 3.8: m_5



a: Plan x-y



b: Plan x-z



c: Plan y-z

Figure 3.9: The path of the global motion of the liver.

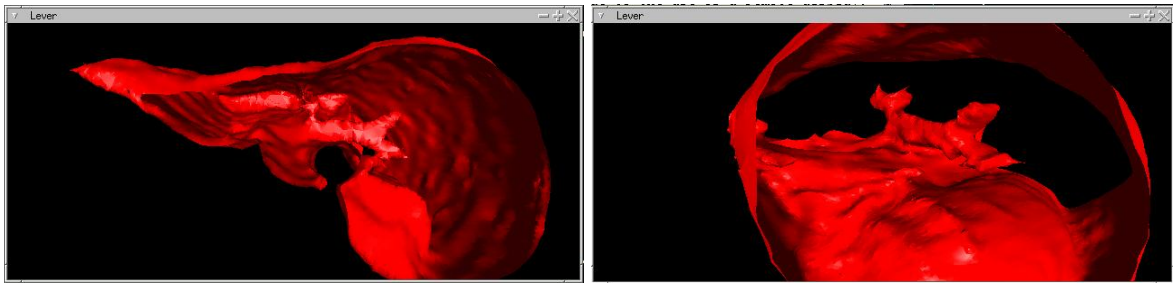


Figure 3.10: Inner structure of the liver.

3.4 Local deformations

By successively apply the non-registration algorithm on I_0 through I_5, we got the transformations on each vertex coordinate of the geometric model respective to the adjacent geometric model. To get an intuitional understanding of the areas where the liver has most deformations, we divide the transformations in five groups:

1). the magnitude of the transformations ≤2mm;

2). the magnitude of the transformations >2mm and ≤5mm;

3). the magnitude of the transformations >5mm and ≤10mm;

4). the magnitude of the transformations >10mm and ≤20mm;

5). the magnitude of the transformations >20mm.

We put different colors on the vertex coordinates to distinguish different transformations. The results are shown in Figure 3.4.1- Figure 3.4.6.

Deformation <= 2mm

Deformation >2mm and <=5 mm

Deformation >5mm and <=10 mm

Deformation >10mm and <=20mm

Deformation >20mm

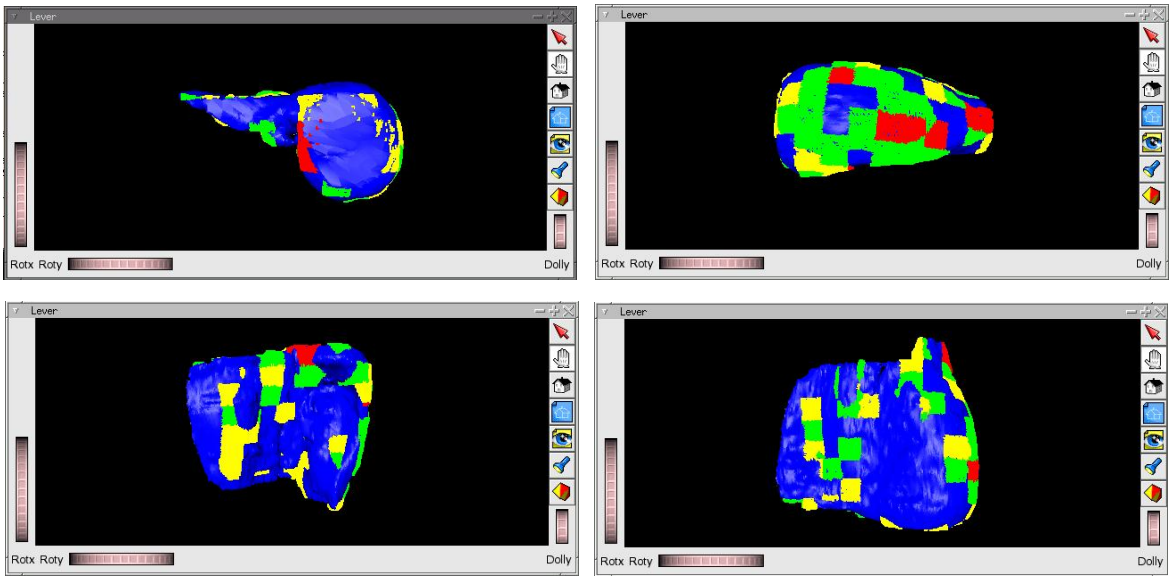


Figure 3.11: Deformation I_0-I_1.

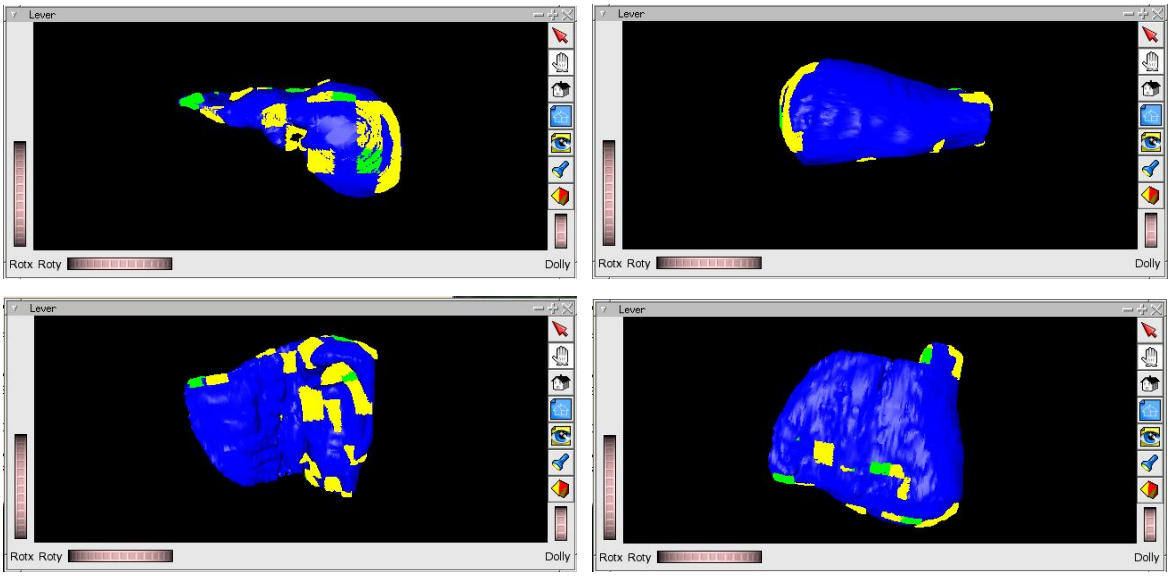


Figure 3.12: Deformation I_1-I_2.

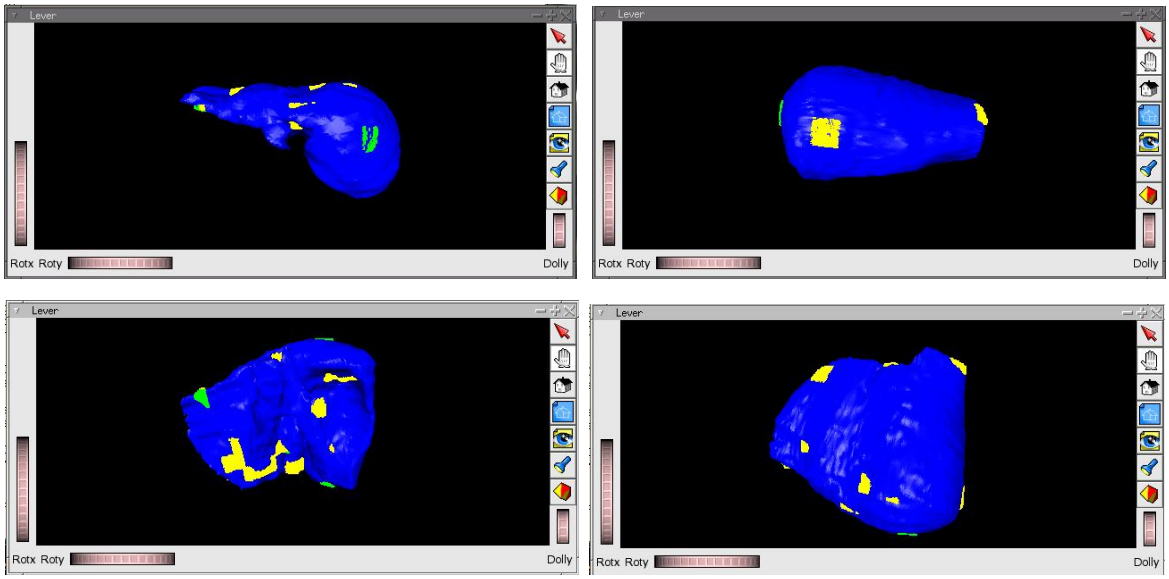
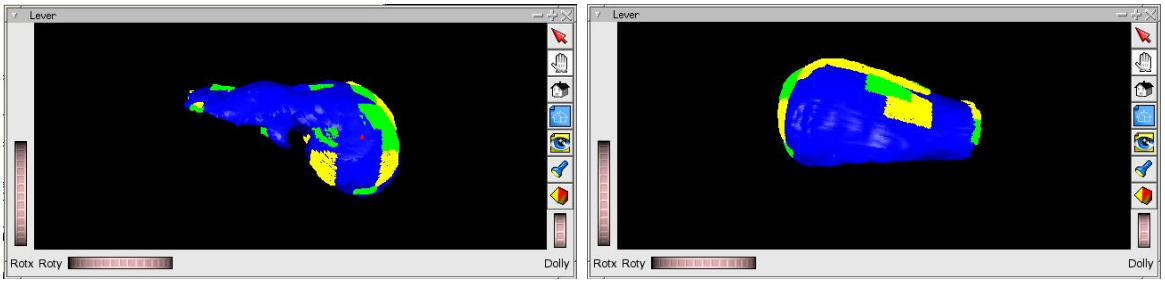


Figure 3.13: Deformation I_2-I_3.



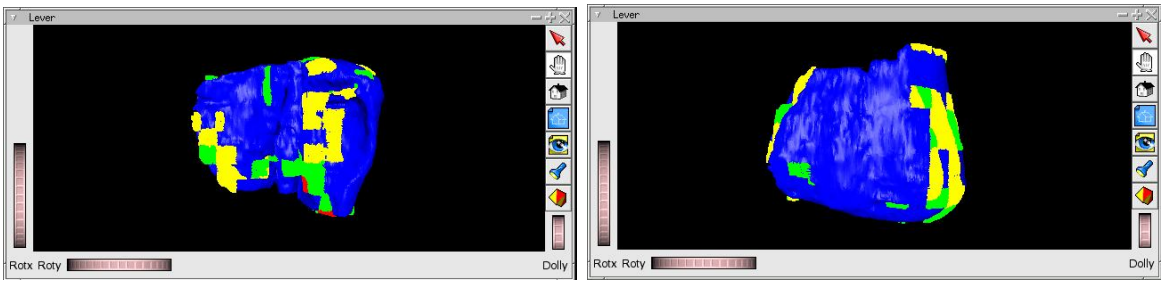


Figure 3.14: Deformation I_3-I_4.

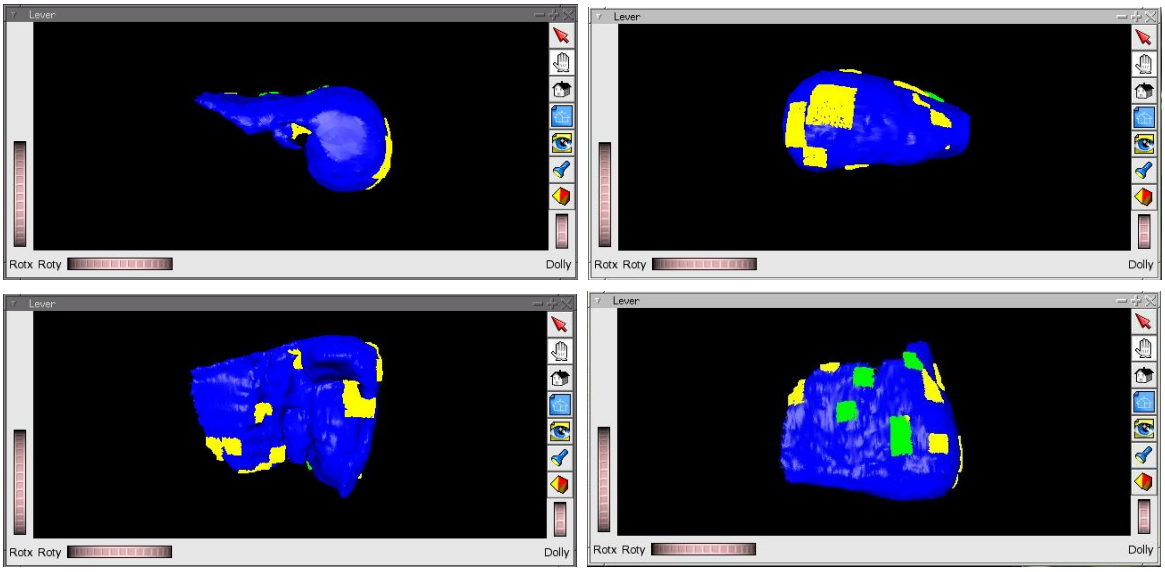


Figure 3.15: Deformation I_4-I_5.

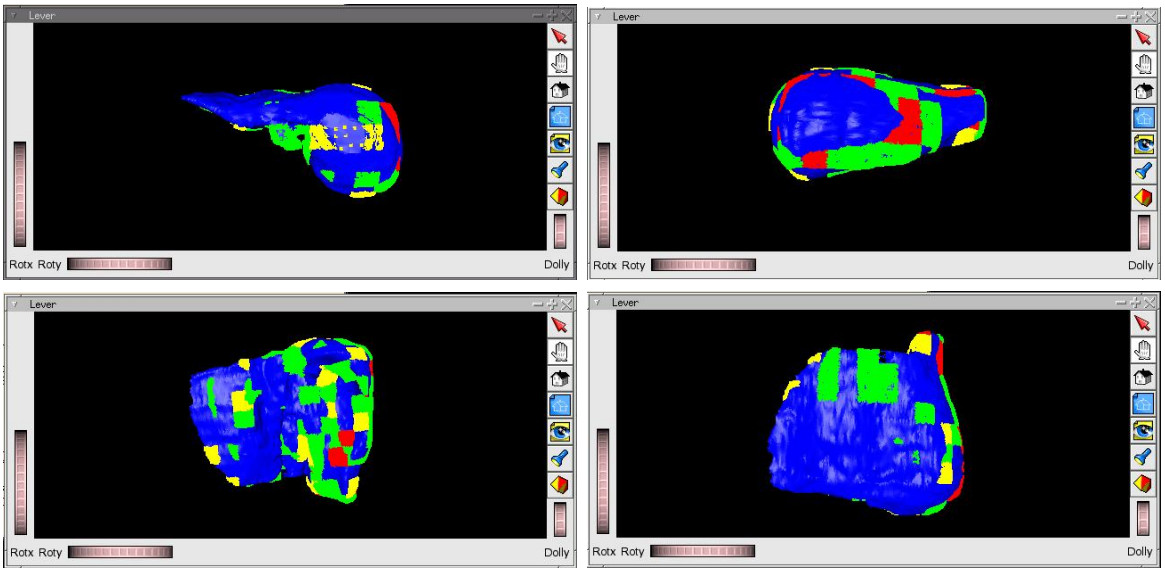


Figure 3.16: Deformation I_5-I_0.

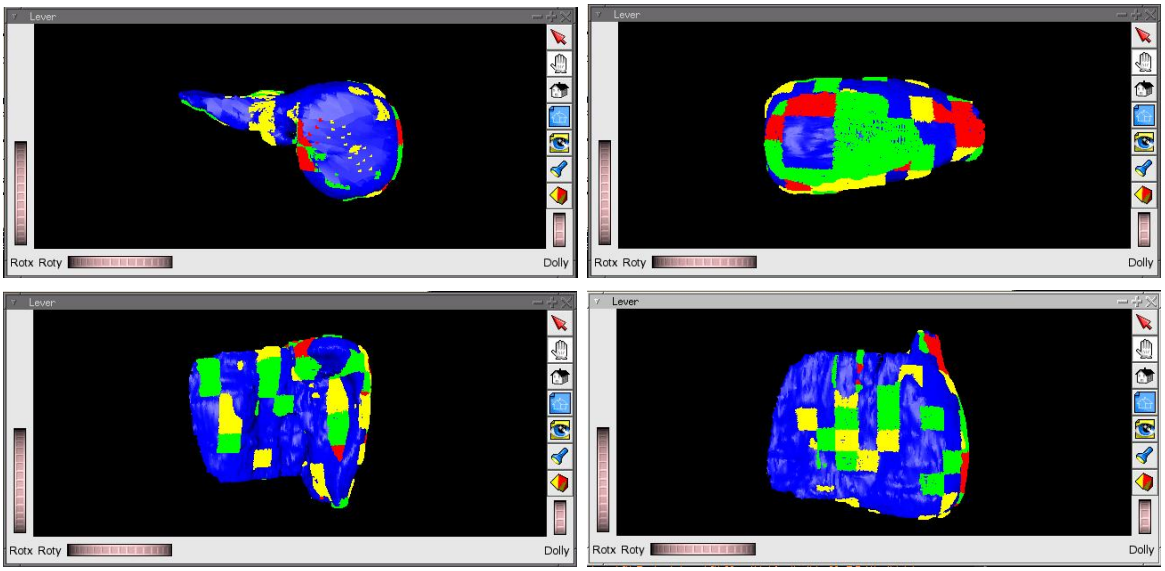


Figure 3.17: Deformation I_0-I_3.

Since we can extract the number of nodes in every geometric model (chapter 2.5.2), we calculated ratio of deformed points in every group versus the total number of points:

	I_0-I_1	I_1-I_2	I_2-I_3	I_3-I_4	I_4-I_5	I_5-I_0	I_0-I_3
	69.25%	**86.86%**	**94.90%**	**78.20%**	**89.47%**	**65.79%**	**68.52%**
	11.92%	**11.76%**	**4.64%**	**12.69%**	**8.24%**	**12.12%**	**10.76%**
	15.11%	**1.38%**	**0.46%**	**8.45%**	**2.16%**	**18.89%**	**16.46%**
	3.71%	**0**	**0**	**0.66%**	**0**	**3.20%**	**4.26%**
	0	**0**	**0**	**0**	**0**	**0**	**0**

Table 3.9.

Chapter 4

Discussion

By observe the global movement of the liver from Table 3.2-Table3.7, we find that the dominant motions of liver is in the S-I direction. From our study, as shown in Figure 3.1, the liver moves downward during inspiration, but has an unstable movement during expiration. By referring the process of respiration, the lung increase in volume with air during inspiration, and the diaphragm moves downward to make space for the lung. When the air is releasing from the lung during expiration, the diaphragm begins to move upward. Since the liver is the organ in the abdomen, which is near the diaphragm, it should has the same movement in the S-I direction as the diaphragm does. In addition, early researchers concluded also that the liver moves downward during inspiration and moves upward during expiration. So our sampling data in I_4 is possible wrong. The reason can be that the six sampling data are not from one respiratory cycle. Our volunteer did a new breath every time he was scanned. It could be difficult for him to breathe in and out the same volume of air every time and suspend his breath at the accurate point. So in the discussion here, we are going to ignore the state I_4. Another obvious translation from our results is in A-P direction. The liver moves forward during inspiration and move backward during expiration. For the subject in this study, the magnitude of the S-I translation between full expiration and full inspiration is about 33mm. The translation in A-P direction is about 17mm and in R-L direction is about 4mm. And there is no rotation result from the rigid-registration by using “3D Slicer”. By comparing our results with the early researches (Table 4.1), our results are very like the other results, the most translation of liver is in the S-I direction and the smallest translation is in the R-L direction. But from Table 4.1, we can see that all our results are little larger than most of the other results. The reason can be that our volunteer did a full inspiration and expiration every time he was

scanned, but the results from early researches were gotten under the normal respiration.

	My study	Rohlfing	Wong	Shimizu
S-I (mm)	33	12-26	32-44	21
A-P(mm)	17	1-12	0-16	8
R-L(mm)	4	1-3		9

Table 4.1: Comparing the results with other researches.

By observing the volume of liver in Table 3.8 and Figure 3.2, we found that the volume of liver varies during respiration. The volume of the liver decreases during inspiration and increases during expiration. As mentioned before, the volume of the abdomen contracts when the lung increases its volume with air under the inspiration, and increases its volume when the lung contrasts under the expiration. Since the liver is in the abdomen, the volume variation of the liver may also accompany with the volume variation of the abdomen.

By observe the results form non-rigid registration, the deformations in relative to the global motions are very small. All deformations are smaller than 20mm. From table 3.9 we can see that those place where have deformations larger than 10mm are only distributed in less than 5% of the surface of the liver. These transformations are distributed in the inferior surface of the liver and the right surface of the liver. This is because the inferior surface of the liver is in contact with the back of the body cavity. When the abdomen contracts during inspiration, it compressed against the back of the body. The right surface of liver is in contact with the rib. During expiration, the liver moves upward and increases it volume, but the increasing is forced by the rib. This causes the deformation in the right surface of the liver. There is also some deformation around the large blood vessels in the liver. Regions around the inferior vena cava, particularly where it passes through the diaphragm, or at the level of the main portal vein may experience compression and thereby expel blood, closing

the vessels to a certain extent. And by comparing with the global transformation, we found that there is direct proportion between global transformation and non-rigid deformation.

Since we only have one study subject, it is very difficult to predict the general deformations of liver by only one set of results. The future work is to study several subjects, which are from different age groups. And we are going to use the Total Lung Capacity (TLC), (the volume of air contained in the lungs at the end of a maximal inspiration) and the Residual Volume (RV) (the volume of air remaining in the lung after a maximal expiration) to control the respiration of the volunteer and get the correct sampling data we want.

Chapter 5

Conclusion

This thesis describes the motion of the liver during respiratory cycle, by segmentation, registration, visualization and volumetric analysis. The results proved that the motion of the liver includes both rigid transformations and non-rigid transformations. The rigid transformations are the dominated transformations. Although the non-rigid transformations is relatively small, but the deformation in some place of the liver can over 1mm, this can also not be neglected.

Reference:

[1] http://en.wikipedia.org/

[2] Hope W. Korin, Richard L. Ehman, Stephen J. Riederer, Joel P. Felmlee, Roger C. GrimmRespiratory kinematics of the upper abdominal organs: A quantitative study Magnetic Resonance in MedicineVolume 23, Issue 1 , Pages 172 - 178

[3] Rohlfing, T, Maurer, Jr. CR, O'Dell WG, Zhong J. "Modeling liver motion and deformation during therespiratory cycle using intensity-based free-form registration of gated MR images". SPIE Medical Imaging Conference Proceedings 2001; 4319:337-348.

[4] J. M. Balter, K. L. Lam, C. J. McGinn, T. S. Lawrence, and R. K. Ten Haken, "Improvement of CT-based treatment-planning models of abdominal targets using static exhale imaging," Int. J. Radiat. Oncol. Biol. Phys. 41, 939–943 (1998).

[5] K.H. Wong, J.W. VanMeter, S.T Fricke, C.R. Maurer Jr., K. Cleary "MRI for modeling of liver and skin respiratory motion". Computer Aided Radiology and Surgery 2004.

[6] Shinichi Shimizu, Hiroki Shirato, Bo Xo, Kenji Kagei, Takeshi Nishioka, Seiko Hashimoto, Kazuhiko Tsuchiya, Hidefumi Aoyama and Kazuo Miyasaka "Three-dimensional movement of a liver tumor detected by high-speed magnetic resonance imaging" Department of Radiology, Hokkaido University School of Medicine, North-15 West-7, Kita-ku, Sapporo, Japan060-8638.

[7] James M. Balter Ph.D., Randall K. Ten Haken Ph.D., Theodore S. Lawrence M.D., Ph.D., Kwok L. Lam Ph.D. and John M. Robertson M.D. "Uncertainties in CT-based radiation therapy treatment planning associated with patient breathing" Department of Radiation Oncology, University of Michigan, Ann Arbor, MI, USA

[8] Eigil Samset "MRI-Guided Interventions Technological solutions"

[9] William G Bradley **"Fundamentals of MRI"**

[10] http://physiol.umin.jp/resp/resp_cycle/

[11] http://www.drstandley.com/bodysystems_respiratory.shtml

[12] Computed Tomography in Collections: Protocol of Application

[13] Mustafa Deha Turan Suleyman "Development of biological voxel-based computational models from DICOM files for 3-D electromagnetic" Demirel University, Faculty of Engineering and Architecture, Department of Electronics & Communication Engineering, 32260, Cunur, Isparta, Turkey

[14] Wei, Kaiping; Zhang, Tao; Shen, Xianjun; Liu, Jingnan "An Improved Threshold Selection Algorithm Based on Particle Swarm Optimization for Image Segmentation". Natural Computation, 2007. ICNC 2007. Volume V. Third International Conference on Volume 5, 24-27 Aug. 2007 Page(s):591 - 594 Digital Object Identifier 10.1109/ICNC.2007.226

[15] William E. Lorensen. Harvey E. "MARCHING CUBES: A HIGH RESOLUTION 3D SURFACE CONSTRUCTION ALGORITHM" Cline General Electric Company Corporate Research and Development Schenectady Publication Date: 1987

[16] http://en.wikipedia.org/wiki/Marching_cubes

[17] http://www.polytech.unice.fr/~lingrand/MarchingCubes/accueil.html

[18]http://www.uio.no/studier/emner/matnat/ifi/INF3320/h06/undervisningsmateriale/lecture6.pdf

[19] http://en.wikipedia.org/wiki/Triangle_strip

[20] http://local.wasp.uwa.edu.au/~pbourke/dataformats/mdl/chunks/trnglStr.html

[21] Tomas Akenine-møller, "Real-time rendering"

[22]http://www.uio.no/studier/emner/matnat/ifi/INF3320/h06/undervisningsmateriale/lecture5.pdf

[23] http://www.vtk.org/

[24] http://www.coin3d.org/

[25] SlicerTraining1 Loading and Viewing Data

[26] Fookes, C.; Williams, J.; Bennamoun, M.; "Global 3D rigid registration of medical images" Image Processing, 2000. Proceedings. 2000 International Conference on Volume 2, 10-13 Sept. 2000 Page(s):447 - 450 vol.2 Digital Object Identifier 10.1109/ICIP.2000.899444

[27] Cha Zhang and Tsuhan Chen "EFFICIENT FEATURE EXTRACTION FOR 2D/3D OBJECTS" Dept. of Electrical and Computer Engineering, Carnegie Mellon University 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

[28] http://www.ascension-tech.com/news/press_060205.php

[29] Ramesh, N. Yoo, J.-H. Sethi, I.K. "Thresholding based on histogram approximation" Dept. of Comput. Sci., Wayne State Univ., Detroit, MI; Publication Date:Oct1995 Volume:142, Issue:5 On page(s):271-279 ISSN: 1350-245X

[30] http://www.rsierra.com/DA/node10.html

[31] Rick Parent "Computer animation algorithms and techniques".

[32] William M Hsu "Direct Manipulation of Free-Form Deformations".

[33] INF-MAT5340

[34] http://www.itk.org/Doxygen/html/classitk_1_1BSplineDeformableTransform.h